AD-A108276

(12) LEVEL II

DTIC
S ELECTE
D

Semiannual Technical Summary

Reconfigurable VLSI Program

31 March 1981

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

Approved for public release; distribution unlimited.

DDC

81 12 08 220

TR-81-153

(12) LEVEL II

Semiannual Technical Summary

Restructurable VLSI Program

31 March 1981

Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-80-C-0002 by

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS

Approved for public release; distribution unlimited.

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-80-C-0002 (ARPA Order 3797).

This report may be reproduced to satisfy needs of U.S. Government agencies.

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Raymond L. Loiselle

Raymond L. Loiselle, Lt.Col., USAF
Chief, ESD Lincoln Laboratory Project Office

Non-Lincoln Recipients

PLEASE DO NOT RETURN

Permission is given to destroy this document when it is no longer needed.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

RESTRUCTURABLE VLSI PROGRAM

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

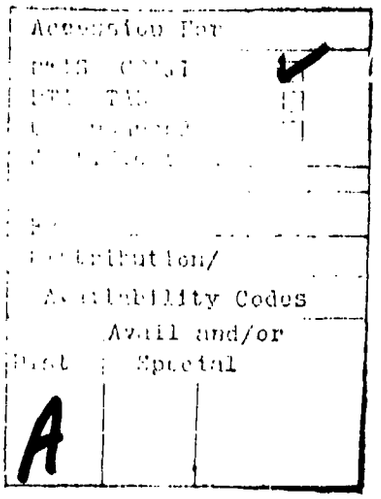
1 APRIL 1980 – 31 MARCH 1981

ISSUED 14 OCTOBER 1981

LEXINGTON MASSACHUSETTS

ABSTRACT

This report describes work on the Restructurable VLSI Research Program sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency during the two semiannual periods, covering 1 April 1980 through 31 March 1981.



CONTENTS

RESTRUCTURABLE VLSI PROGRAM

I. PROGRAM OVERVIEW AND SUMMARY

The main objective of the Lincoln Restructurable Very Large Scale Integration (RVLSI) Program is to develop methodologies for designing whole-wafer-scale systems with complexities approaching a million gates. In our approach, we envisage a modular style of architecture comprising an array of cells embedded in a regular interconnection matrix. Ideally, the cells should consist of only a few basic types. The interconnect matrix is a fixed pattern of metal lines augmented by a complement of programmable switches or links. Conceptually, the links could be either volatile or nonvolatile. They could be of an electronic nature such as a transistor switch, or could be permanently programmed through some mechanism such as a laser. The present main thrust of the current RVLSI Program is toward laser-formed interconnect.

The link concept offers the potential for a highly flexible, restructurable type of interconnect technology which could be exploited in a variety of ways. For example, logical cells or subsystems found to be faulty at wafer probe time could be permanently excised from the rest of the wafer. The flexible interconnect could also be used to "jump around" faulty logic and tie in redundant cells judiciously scattered around the wafer for this purpose. Also, the interconnect could be tailored to a specific application in order to minimize electrical degradations and performance penalties caused by unused wiring.

Further, the testing of a particular logical subsystem buried deep within a complex wafer-scale system is a very difficult problem. A properly designed restructurable interconnect matrix could be temporarily configured to render internal cells both controllable and observable at the wafer periphery. In this way, each component cell or a tractable cluster of cells could be tested in a relatively straightforward way using standard techniques. It is possible to provide this capability, though to a somewhat limited degree, with the laser programmed type interconnect approach.

With an electronic type of linking mechanism it is possible to think in terms of a dynamically reconfigurable system. Such a feature could be used to alter the functional mode of a system subject to changes in the operating scenario, or it could be used to support some degree of fault tolerance if the system architecture were suitably designed.

A major near-term goal for the RVLSI Program is to develop computer-aided design automation to exploit the power and flexibility of the restructurability concept, specifically in terms of laser-formed connectivity. A conceptual diagram of the type of system envisioned is shown in Fig. I-1. From an initial concept, a system-level design is decomposed or partitioned into a number of basic cell types and interconnections of the cells. Design automation will expedite the task of cell design; make decisions on the degree of redundancy required for a given cell complexity, processing technology, and desired wafer yield; optimize the distribution of redundant resources across the wafer with respect to expected defect producing mechanisms; and allocate the amount of interconnect capacity necessary to assure that functional cells can be connected with a preassigned probability of success. The cell-design automation will take into account the debug/testability problem, will help develop test strategies, and will provide simulation tools and extra hardware mechanisms as needed to enhance testability.

Once a wafer design has been formulated and checked at the mask level for consistency with processing technology design rules, it can be forwarded to fabrication. Upon return, it must be subjected to first-level testing which will verify the state of the interconnect matrix and individually test each of the component cells. Each wafer, along with its unique defect map, will be forwarded to further automation which marries the defect data with physical topological (layout) information and logical interconnect requirements. During this phase of the operation, logical cells will be mapped onto functional resources (assignment) and connection paths among them will be defined (linking). During the final phase, a laser/probe facility will be driven by the connectivity data and actually program the necessary links. Feedback from the laser assembly will allow monitoring of the link programming process and provide the opportunity to reiterate the assignment/linking process should occasional faulty links and/or interconnect be discovered along the way.

Given this global perspective, several major areas of research can be identified in the context of the restructurable VLSI concept:

(a) System architectures and partitionings for whole-wafer implementations.

(b) Placement and routing strategies for optimal utilization of redundant resources and efficient interconnect.

(c) Assignment and linking algorithms to exploit redundancy and flexible interconnect.

(d) Methods for expediting cell design with emphasis on functional level descriptions and enhanced testability.

(e) Methods for testing complex, multiple-cell, whole-wafer systems.

Complementary work on the development of various link and interconnect technologies as well as fabrication/processing technology is being supported by the Lincoln Air Force Line Program, and results are reported under the Lincoln Laboratory Advanced Electronic Technology Quarterly Technical Summary.

Work for this period is reported under the general headings of RVLSI Technology (Sec. II), Design Aids for RVLSI (Sec. III), and RVLSI Testing and Applications (Sec. IV). In the area of RVLSI Technology, a unique approach to a limited style of restructurability was developed and verified through MOSIS. Termed "dynamic bonding," this approach connects all subsystems of a die to a common set of I/O pads through an internal bus. Each system becomes visible to the outside world by enabling its power pin. Thus, each subsystem can be connected to the outside world in succession for testing or operation, thereby avoiding a unique bond-out. Also, a standardized frame packaging technique has been developed for MOSIS as an alternative to the MPC style of Mead and Conway. The circuit designer is required to fit his design into one of four standard fractional die sizes. This simplifies the bonding and die probing problem and improves packaging economics.

Some experiments have been conducted exploring the possibility of fabricating laser-programmable links within the MOSIS NMOS process. Although these links are known to perform well when precise control over processing technique and materials is possible, the problem is more difficult when this is not the case.

Under the heading of Design Aids for RVLSI, a number of CAD tools for cell design have been considered and are at various stages of development. A simple, program-based, mask-level

layout system called LICL has been developed and is operational on the PDP-11/70 and VAX 11/780 facilities. Two higher-level design systems are in progress which seek to exploit regularity in structural topology. GAMMA represents an attempt to automate the "dense gate matrix" discipline of Lopez and Law. MACPITTS is a system which accepts functional level descriptions and permits rapid design of cells which can be expressed as combinations and interconnections of a small number of regular, elemental logic structures (e.g., finite state machines and register arrays). The structure of MACPITTS allows the designer to take advantage of natural concurrency in a given design which is an asset for signal-processing applications. Since only a limited number of regular building blocks is permitted, the potential for improved testability, fault tolerance, and performance simulation tools is offered.

Beyond the cell-design problem, considerable effort has been expended in the areas of placement, routing, assignment, and linking for whole-wafer systems. An experimental system was developed to study such issues as redundancy and connectivity requirements as a function of cell yield, assignment and linking algorithm behavior and computation requirements, and the impact of imposing various types of regularity constraints on topological interconnect structure. A subsequent, production-oriented assignment/linking/laser programming system is well under way for the packet radio integrator application.

With respect to testing, some preliminary analyses have been conducted examining possible approaches to enhancing cell testability assuming an external tester approach. The established "stuck-at" fault detection approach is shown to have a number of undesirable weaknesses, and the Boolean difference technique is suggested as an alternative with better properties but requiring a larger test vector base. Also, a design has been developed for a logic testing element and a concomitant testability criterion ideally suited to dynamic circuit applications and appropriate for automatic computer control. The tester can absorb a test vector stream supplied by a host, apply it at speed to a device-under-test, and record results. The device supports a limited looping capability and can apply "hold" conditions. These devices can be chained together, 4 bits to a slice, to provide as wide a stimulus/response interface as desired. Although intended in the near term as the elements of a standalone system to aid in cell testing, they can be considered as a specialized cell type unto themselves which could be scattered among "payload" cells in a whole-wafer system to enhance global testability.

Also, a methodology has been developed for checking the functionality of a wafer-scale interconnect matrix wholly independent from the cells themselves. This method requires only a minute amount of extra interconnect over and above what is normally needed and can be automated in a very straightforward way. The test algorithm is seen to be very rigorous for its conceptual simplicity and has been adopted for the packet radio integrator.

Applications addressed to date include an integrator for packet radio application, a filter for speech-processing applications, and an FFT system for vocoders. The integrator effort is progressing in two phases. During the first phase, a scaled-down version is being developed using the Lincoln CMOS gate array as a cell design expedient. This will permit evaluation and refinement of the assignment/linking/laser programming automation in a limited, controlled manner. The final design will be developed during the second phase and will require modifications, extensions, and enhancements of the CAD tools already in place.

The speech filter was used as a primary vehicle for debugging the LICL layout system and has been processed through MOSIS. It is currently undergoing evaluation.

A design study was conducted investigating DFT structures suitable for both speech-oriented applications and implementation via the multi-cellular RVLSI approach. Several architectural approaches were examined, and it was concluded that the pipeline FFT is best matched to the RVLSI philosophy. Butterfly, memory and commutator functions were considered as candidate elemental cell types.

Finally, the VAX 11/780 CAD tool development facility is operational and is in the process of being enhanced with extra disk space, an ARPANET connection, a color display, and a link to the laser assembly. LICL, MACPITTS, and a flexible design-rule checker are being installed. The production-oriented assignment/linking/laser programming automation for the packet radio integrator is also being implemented on this facility. Further, the Lincoln wafer probe facility has been used successfully on five MPC runs thus providing a quick-turnaround service to the ARPA MOSIS community.

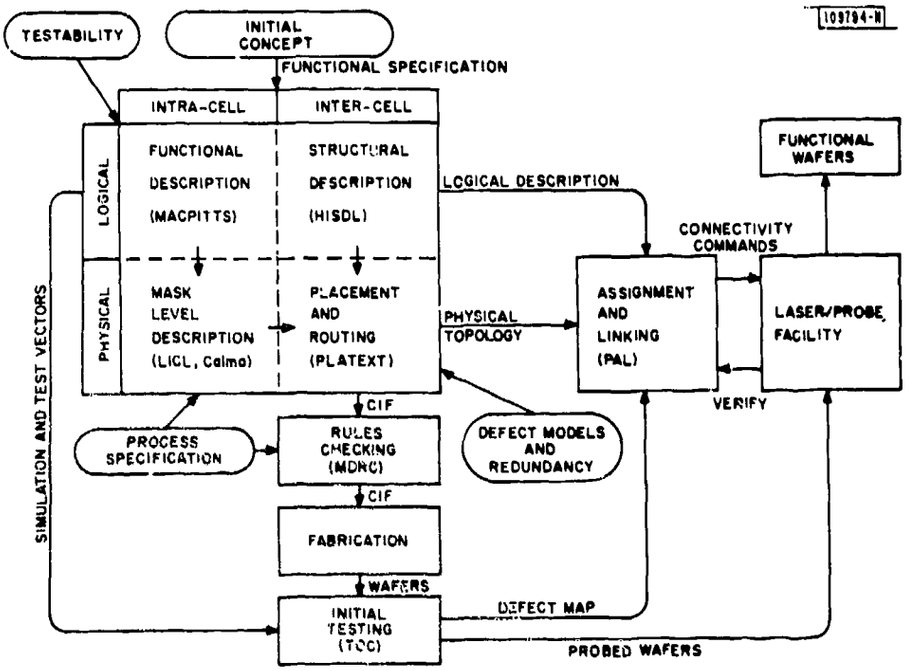


Fig. I-1. Restructurability via laser-formed connectivity.

II. RVLSI TECHNOLOGY

A. DYNAMIC BONDING FOR MULTIPROJECT CHIPS

1. Introduction

In the Multiproject Chip (MPC) style of fabricating experimental NMOS integrated circuits, many unrelated designs, each with its own I/O pads, are placed on a single die. The multiproject die is packaged in a 40-pin or larger package, with one project bonded to the package pins. Figure II-1 shows a 255 × 303-mil die from MPC79 (Ref. 1) inside an array of package bonding pads. MPC79 included 82 projects on 12 die types spread over two wafers.

The MPC implementation technique tends to minimize mask and wafer fabrication cost per design, but not without penalty. Disadvantages include the following:

(a) Wafer probe testing is not feasible;

(b) The large number of different bonding maps results in costly packaging;

(c) Only one project per die can be tested.

A method is described here for "dynamic bonding" by which it is possible to connect each project, one at a time, to a standard array of die pads. The method can be extended to permit communication between projects, allowing designers to assemble individual projects into a single-chip system[2] thus demonstrating some of the advantages of Restructurable VLSI[3,4] techniques.

2. Assumptions for Dynamic Bonding

The following assumptions have been made regarding dynamic bonding:

(a) Dynamic bonding should have small impact on the design process; in particular, the designer should not be unduly constrained in choice of aspect ratio for the design or apportionment of total allowed I/O between input and output.

(b) A package pin should logically look like a project pad, and a tri-state capability should be provided.

(c) Dynamic bonding should be implementable in the current silicon foundry standard NMOS process.

(d) The probability that a mistake by one designer makes an entire die inoperable should be minimized.

3. Proposed Technique

We propose that a set of bonding pads, each with associated I/O circuitry, be located on the periphery of each die. Projects are placed on the die and their circuits connected to the die I/O pads via an on-chip bus as depicted schematically in Fig. II-2. The projects and the die I/O pads share a common ground but are powered by separate V_{DD} lines. A project is selected (connected to the die I/O pads) by supplying power to its V_{DD} line (die I/O pads are always powered). The V_{DD} line of unselected projects is grounded (or left unconnected).

Due to our desire not to constrain the designer in I/O assignment, and to simplify routing of interconnect, each Data line of the on-chip bus may be used for either input or output. This means that all project output drivers must be high impedance when the project is not selected,

and that the selected project must be able to specify whether a die I/O pad is used for input or output. To achieve this, each Data line is accompanied by a Select line which determines the mode of operation of the die I/O pad. Only selected projects are allowed to affect the state of Select lines.

Project I/O circuits are provided as a library cell in several versions. The three most important of these cells are depicted in Fig. II-3. The three cells are shown connected to the same pair of Data (D) and Select (S) lines, and are thus presumably parts of different projects. The cells are designed such that they appear as a high-impedance load to the on-chip bus when not powered. The operation of each is described briefly below.

The Project Input cell contains a bus receiver circuit for inputting data from the bus, and a pull-down transistor that sets the Select line to logical 0 when powered. The Project Output cell contains a bus driver circuit for outputting data to the bus, and a pull-up transistor that sets the Select line to logical 1 when the circuitry is powered. The Project Tri-State cell is the most complex of the three. First, it contains a bus driver circuit for setting the state of the Select line as determined by the enable line from the project. Second, a tri-state bus driver is employed to drive the Data line, thus allowing data to be input from this bus line as well as output. Finally, a bus receiver circuit provides the clean interface between the Data line and the project. Note that the existence of the bus receiver circuits in the Project Input and Project Tri State cells minimizes the probability that a project design error can adversely affect the operation of the on-chip bus for other projects.

The block diagram of the die I/O pads is given in Fig. II-4. Each pad is associated with one pair of Data and Select lines. The pad circuitry is bidirectional – it may be used to drive signals from the outside world to the on-chip bus or, alternatively, to drive signals from the on-chip bus to the outside world. The state of the Select line determines the mode of operation.

4. Advantages and Disadvantages of Dynamic Bonding

The primary benefit of the dynamic bonding scheme is the standardization of the die bonding pads. This standardization makes feasible testing by wafer probing and reduces the cost of packaging. In addition, since all projects on a single die may be tested after packaging, fewer packaged chips may be required.

Dynamic bonding can be readily extended to allow communication between projects. Thus, projects would be first tested as separate entities, and then, if found operable, would be connected together into a single-chip system. These single-chip systems could conceivably include redundant components and thus demonstrate the advantages of restructurability.

Project selection via the power pin requires no additional complexity in the project and helps to ensure that a design in one project does not kill the die. However, the requirement of separate V_{DD} pads reduces the total number of available data pins. For a 40-pin package and N projects on a die, the maximum available logic pins will be 40-2-N.

Because of the required interconnect area and die bonding pads, fewer projects can be packed on a die than at present. A packing experiment was performed in which each project in MPC79 was increased in size enough to allow placement of interconnect metal lines for that project's pads on two sides of the project. Allowance was not made for die pads, so the results may be slightly optimistic. The 82 projects were placed on 23 dice as compared with 12 for MPC79 or, excluding one large project which required an entire die, 22 and 11. The maximum number of projects on a die was 6.

The dynamic bonding interconnect lines must be routed on the die. In the MPC implementation style, this design task would be performed by the data management contractor. Alternatively, groups of designers could take advantage of dynamic bonding by submitting whole-die projects already routed. In many cases, where projects have relatively few I/O terminals, some package pins need not be shared.

Finally, due to delays on the on-chip bus, a dynamically bonded project should be expected to run somewhat slower than its current MPC counterpart. These delays are minimized by the design of appropriately sized bus drivers.

5. Dynamic Bonding Circuits

The I/O circuits needed for dynamic bonding (Figs. II-3 and II-4) were designed for MOSIS implementation and placed in a cell library for use in a test project. The on-chip bus driver circuits are two-stage superbuffers with a scaling of transistor sizes according to the guidelines of Mead and Conway.[5] The requirement that bus driver outputs assume a high-impedance state with the power off is achieved by using an enhancement-mode pull-up transistor at the output of the stage. This means that logical 1's on the bus will be a threshold voltage below V_{DD}. The output transistors, which have a width/length of 24, are designed to drive a worst-case interconnect run equal in length to the sum of the length and width of the MPC79 die (14.2 mm). The capacitance of such a run is estimated as 2.5 pF.

The bus receiver circuit is a one-stage superbuffer which provides isolation between project circuitry and the on-chip bus and restores logical 1 signals to V_{DD}. The pad driver circuit is from the standard MPC (MOSIS) cell library.

The project I/O circuits were designed to be completely interchangeable with existing cells, both electrically and geometrically. Thus, a project designer could convert his design to a dynamic bonding implementation simply by replacing the existing I/O cells with the dynamic bonding I/O cells. These cells are available in CIF (Caltech Intermediate Form).

6. The Dynamic Bonding Test Chip

A test chip was designed so that important I/O timing paths could be characterized for both the existing cells and the dynamic bonding cells (see Fig. II-5 for block diagram). The PadIn, PadOut, and PadTri-State circuits are from the MPC (MOSIS) cell library. The Project Input, Project Output, and Project Tri-State circuits are the dynamic bonding circuits. The I/O Pad circuits are the dynamic bonding die pads. Finally, the Logic block was designed to allow measurement of various timing paths in the Tri-State circuits, as described below.

The dynamically bonded portion of the test chip contains two mini-projects, powered by V_{DD1} and V_{DD2}. Each mini-project contains two I/O terminals, thus requiring two pairs of on-chip bus lines (D_0-S_0 and D_1-S_1). For purposes of this experiment, two sets of die I/O Pads have been included, A and B, powered respectively by V_{DD3} and V_{DD4}.

When I/O Pads A are active, the bus connecting die pads to project I/O circuits is short (low capacitance). When I/O Pads B are selected, a worst-case bus is simulated by loading the ends with large depletion-mode transistors with gate capacitances of 2.5 pF. In addition, a portion of the bus between the two sets of die I/O Pads has been replaced with polysilicon segments of 2K ohms. These segments serve to simulate the effect of bus crossovers as well as isolate fast transitions on the best-case bus from the large capacitance of the worst-case bus.

The principal timing measurement to be made is the data path delay – the delay between a project output and corresponding chip output. For the standard I/O scheme this is simply the delay of a pad driver, and can be measured using project terminals IN_3 and OUT. Tri-State circuits not only have a delay in the data path, as above, but also a delay between enabling the circuit and the presence of valid data at the output. The timing delays associated with the Tri-State circuits are measured with the assistance of the Logic circuitry (Fig. II-6) and two control pads (IN_1 and IN_2). Control pad IN_1 determines which of the two Tri-State circuits is enabled. Only one Tri-State circuit is enabled for output at a time, the other is used as an input. Control pad IN_2 determines what each Tri-State receives as input. When IN_2 is zero, $D_{IN_1} = D_{IN_2}$, and $D_{IN_2} = D_{OUT_1}$. When IN_2 is one, $D_{IN_1} = 1$ and $D_{IN_2} = 0$, regardless of D_{OUT_1} and D_{OUT_2}.

The data path delay of the Tri-State circuits is measured with IN_2 set to zero and IN_1 set to either zero or one. Consider the measurement as done on the library Tri-State circuit, PadTri-State. If IN_1 is zero, then TS1 is enabled for output, and the delay between a driven signal at TS2 and the inverted output at TS1 is measured. If IN_1 is one, then TS2 is enabled for output, and the delay between a driven signal at TS1 and the inverted output at TS2 can be measured.

The enabling delay of the Tri-State circuits is measured with IN_2 set to one and TS1 tied to TS2 as a common output node. When IN_1 is zero, TS1 is enabled for output, TS2 is disabled, and the common output node will be driven to one (since the Logic has frozen D_{IN_1} at one). When IN_1 is one, TS2 is enabled, and the output is driven to zero. The enabling delay is measured by driving IN_1 and watching the TS1-TS2 output.

Note that, in all the above timing measurements, the output of the driven signal is inverted as well as delayed. This means that the output may be tied to input in the manner of a ring oscillator, and the timing measurements may be made without the benefit of a signal generator.

The timing delays associated with the dynamic bonding circuitry are measured using the methods described above. The data path delay of the Project Input and Project Output circuitry is measured by turning on V_{DD2} and powering V_{DD3} (to use best-case bus) or V_{DD4} (for worst-case bus). The delays associated with the Project Tri-State circuitry are measured by powering V_{DD1} and powering V_{DD3} or V_{DD4}. When V_{DD3} is powered, the A I/O Pads are selected, and project terminals A0 and A1 are used for the measurements. When V_{DD4} is powered, the B I/O Pads are selected, and project terminals B0 and B1 are used.

7. Experimental Results

The dynamic bonding test project was laid out and the masks described in CIF with the aid of LICL (Lincoln Integrated Circuit Language). The CIF description of the project was submitted to MOSIS on 6 January 1981. Three packaged chips were returned on 12 March. All three chips were functional.

The results of the delay-time measurements described previously may be found in Table II-1. The test strip's 49-stage ring oscillator had a relatively slow oscillation frequency of 11.0 MHz [measurement (0)]. In the past, this frequency has been typically 15 MHz (see Ref. 1). The results of the I/O timing measurements should be considered with this in mind.

There were three groups of four I/O timing measurements made. Measurements (1) through (4) involve the standard I/O circuitry and are used to examine the following:

(a) The data path delay through the PadIn and PadOut circuits,

(b) The data path delay through the Tri-State Circuits of TS1 and TS2, with TS2 being the input pad and TS1 being the output;

(c) Same as (b), with TS1 the input and TS2 the output;

(d) The enabling/disabling time delay for the Tri-State circuits.

The second and third groups of four are the corresponding measurements made for the dyanmic bonding circuitry for the best-case bus [measurements (5) through (8)] and the worst-case bus [measurements (9) through (12)].

The IN_3-OUT loop of measurement (1) oscillated with a period of 26 ns. This loop delay can be separated into three components: PadIn to the inverter input, inverter input to output, and inverter output (PadOut input) to PadOut output. The first of these is assumed to be zero since it is nothing more than a diffused wire. The calculated pair delay (the sum of the low-to-high and high-to-low delays) of the inverter is 6 ns. Therefore, the pair delay of the PadOut circuit is 20 ns. After accounting for the difference in the low-to-high and high-to-low delays of the inverter, it was found that these delays were nearly symmetrical for the PadOut circuit (the output low-to-high delay was 1 ns slower). Thus, the one-way delay of the PadOut circuit (regardless of the direction of the transition) is about 10 ns. Increased loading capacitance was observed to affect the one-way delay at the rate of 0.11 ns/pF.

TABLE II-1
DYNAMIC BONDING TEST PROJECT EXPERIMENTAL RESULTS

	Power	Nodes	IN_1	IN_2	Oscillation Period (Avg.) (ns)
(0)	V_{osc}	19-Stage Ring Oscillator	–	–	90, 92, 90-91
(1)	V_{DD0}	IN_3-OUT	–	–	24, 28, 27-26
(2)		TS1-TS2	0	0	62, 68, 66-65
(3)				0	78, 88, 82-83
(4)		IN_1-TS1-TS2	–	1	134, 142, 136-137
(5)	$V_{DD2,3}$	A0-A1	–	–	62, 64, 63-63
(6)	$V_{DD0,1,3}$	A0-A1	0	0	91, 103, 95-96
(7)		A0-A1	1	0	86, 98, 92-92
(8)		IN_1-A0-A1	–	1	134, 155, 140-143
(9)	$V_{DD2,4}$	B0-B1	–	–	88, 88, 88-88
(10)	$V_{DD0,1,4}$	B0-B1	0	0	117, 128, 122-122
(11)		B0-B1	1	0	111, 120, 116-116
(12)		IN_1-B0-B1	-	1	148, 165, 152-155

The timing loop of measurement (5) can be separated into two (to and from the project) bus driver pair delays, a PadOut pair delay, a bus receiver pair delay, and an inverter pair delay. The result of measurement (1) can be subtracted from (5) leaving 37 ns for two bus driver pair delays, or 16 ns per pair delay. Assuming that the low-to-high and high-to-low delays of the driver are roughly symmetrical (as they were for the PadOut circuit), the one-way delay of the bus driver circuit (driving a best-case bus) is 8 ns.

The effect of increased bus loading can be determined by subtracting measurements (5) from (9), (6) from (10), and (7) from (11) to get 25, 26, and 24 ns, respectively. These results indicate that the pair delay of a bus driver driving a worst-case bus is about 12 ns greater than for the best-case bus. This result is confirmed by subtracting measurement (8) from (12) to get 12 ns directly (in these two measurements inputs to the project arrive via the control pads, not the on-chip bus, and therefore have only one bus driver pair delay per cycle). The one-way delay of the drivers is increased by 6 ns to a total of 14 ns for the worst-case bus.

Because the delays of the Logic block itself were substantially greater than the delays to be measured, it was impossible to do any meaningful extraction of tri-state enabling times from the data. A comparison of the results of measurements (4), (8), and (12) does, however, indicate that the tri-state enabling delay time for the dynamic bonding circuitry suffers roughly the same penalty as does the data path delay.

8. Summary

A scheme for the dynamic bonding of experimental NMOS test projects has been proposed. In the proposed scheme, all projects of a multiproject die are interfaced to the same set of physical I/O pads. Project selection is done simply by turning on the power for the project to be selected. Dynamic bonding would make it feasible to test projects at the wafer level, simplify the packaging procedure, and allow all projects on a die to be tested after packaging.

A test project for the proposed scheme has been designed, fabricated, and tested. The dynamic bonding I/O circuits were 100-percent functional. The project-die pad delays were 8 and 14 ns for the short and long buses, respectively.

The method can be extended to permit communication between projects and provide designers a bus and other support circuitry for connecting their projects into a single-chip system.

B. A STANDARD FRAME STYLE OF PROTOTYPE IC FABRICATION

1. Introduction

An implementation system for prototype custom integrated circuits should minimize cost per design for mask making, wafer fabrication and packaging, produce chips with a fast turnaround time, and minimize design constraints. The goal of low cost is especially important for student projects. The MPC approach pioneered by Mead and Conway has been successful in meeting these goals.[6] In MPC, a number of designs are packed onto a standard size die and several different die types are assembled on a mask. Each design on a die (or chip) has its own bonding pads, and a single design is bonded out when a chip is packaged.

The MPC approach minimizes mask area at the expense of wafer area since it is easy to replicate the mask set on many wafers. The MPC and MOSIS practice has been to fabricate one set of ten wafers for each mask set. The resulting costs for mask generation and wafer fabrication have been about equal. The number of times a die is repeated on a mask is determined by

how many chips of each design are required from a 10-wafer fab run and how much redundancy is needed against mask defects. With increased confidence in mask making and fabrication, the number of die type replications has been reduced in recent MOSIS runs from the earlier MPC runs. With electron-beam mask making there is no penalty in having many unique patterns on a mask.

Two problems with the MPC scheme are the impossibility of doing wafer probing and the delay and cost of packaging. Both are a result of the unique bonding pad layout of each design. This section describes an alternative implementation scheme which standardizes the pad layout and therefore permits wafer probing and simplifies packaging. Its advantages and disadvantages will be discussed and some comparisons made with the MPC style.

2. Standard Frames or Multiproject Wafer

If designers are encouraged, or required, to use one of a few standard frames with bonding pads then it is possible to do wafer probing with only a few probe cards, and packaging is identical for all designs using the same standard frame. The standard frames may be used in two different ways. First, they may be packed into a standard single-size die which is cut and packaged on one chip, MPC style. Or, each Standard Frame die may be separated from the wafer as a separate chip, in which case we have what might be termed a multiproject wafer. The second method makes more efficient use of the wafers at the expense of a higher scribing cost.

Comparisons of mask area consumption by the Standard Frame vs MPC have been made using data on the projects of MPC79 (Ref. 1) and MPC580 (Ref. 7). The MPC79 die which was used as a standard has an overall size of 7696 × 6477 microns and a usable area of 7548 × 5926 microns. Full-, half-, and quarter-size frames were defined with adequate space for test strips and scribe marks. The project area dimensions for these dice are:

Full	7548 × 5926 microns
Half	5926 × 3700 microns
Quarter	3700 × 2688 microns

The MPC580 run used two mask sets and two die sizes, one slightly larger and one slightly smaller than the MPC79 die. One MPC580 project did not fit on an MPC79 die and is excluded from the comparisons.

Table II-2 summarizes the data and shows the relative mask area for one copy of each type. The SF scheme requires 2.16 and 1.67 times the MPC mask area for MPC79 and MPC580, respectively. However, to obtain a certain number of packaged devices for each design from a 10-wafer fab run, more than one copy of a die type per wafer may be required. Assume that we wish to have a 90-percent probability that each designer receives two packaged chips free of processing defects. Using the yield equation:

$$Y = \{1/[1 + A/(A0 * 4.5)]\} ** 4.5$$

and A0, area per defect, equal to 30 mm^2 the results of Table II-3 are obtained.† Since there are so many small projects, yield for a one-eighth size project has also been calculated. For

† This number for A0 may be optimistic. "Status '80 – A Report on the Integrated Circuit Industry," Integrated Circuit Engineering Corporation, gives A0 = 19.3 mm^2 as a typical industry MOS number. This gives project yields of 0.15, 0.36, 0.61, and 0.78 for the four sizes.

TABLE II-2
MPC AND SF COMPARISONS

	MPC79	MPC580
Number of Projects	82.0	167.0
MPC Die Types	12.0	36.0
Full-Size SF Projects	2.0	10.0
Half-Size SF Projects	12.0	35.0
Quarter-Size SF Projects	68.0	127.0
SF Equivalent MPC Dice	25.0	60.0
SF-to-MPC Mask Area	2.16	1.67

TABLE II-3
YIELDS FOR DIFFERENT SIZE PROJECTS

Project Size	Project Yield	Devices Packaged	Probability of Two Good Devices
Full	0.28	10	0.85
Half	0.51	6	0.90
Quarter	0.73	4	0.93
Eighth	0.85	3	0.94

the assumed A0, these yields are conservative since many projects are smaller than the SF allowed area. These data can be used to determine how many copies of each die type must be placed on the mask to obtain sufficient chips from a 10-wafer fab run. We assume that the individual projects on the SF wafers are all separated. Then, the number of MPC full-size equivalent dice required on the masks is shown in Table II-4 where the first number is the minimum required by the yield calculations and the second provides mask defects. The MPC79 run had 54 project dice plus 3 drop-in test dice on each wafer, so with either MPC or SF one wafer would be used for MPC79 and two for MPC580 (with some loss of mask defect redundancy with SF).

As noted earlier, one complication with the Standard Frame is the necessity to cut several different size dice from one wafer. To facilitate this it would probably be desirable to format the wafer in a regular fashion, such as having only one size of die in a row. This may increase the number of equivalent dice for SF in Table II-4.

TABLE II-4
NUMBER OF FULL-SIZE EQUIVALENT DICE ON MASK

	MPC79	MPC580
MPC	33, 34	77, 81
Standard Frame	25, 50	60, 120

The LICL program has procedures for generating full, 1/2, 1/4, and 1/8 size frames with bonding pads and power distribution buses on the periphery. The bonding pad at each position is present, even if not used, so that standard package bonding may be done for each chip. The 1/8-size die has 24 pads and the others 40, with space for additional pads on the two larger ones. Standard power pin locations are present, but the designer could change their position. In LICL, and in other similar systems the desired I/O circuit is placed by name at the desired location. For other systems, a table of pad positions would be required. Two Lincoln Laboratory projects have been built in MOSIS using Standard Frames.

3. Advantages and Disadvantages of Standard Frame

The principal advantage of using the Standard Frame is that the packaging process is standardized and should be less expensive and less error prone than for MPC. It also makes it possible to do wafer probing with a few probe cards – three, for instance, in the above example. As shown in Table II-4, the relative efficiencies of mask and wafer utilization depend on defect redundancy strategies but are about the same. The SF style yields more dice than required for the small projects.

The principal disadvantage of the SF style is the constraint that a design must fit within one of the Standard Frames. Note, however, that the comparisons made here have been done with a set of designs for which the only constraint was a maximum size, so it may not be necessary to emphasize the use of a smallest possible frame to achieve efficiencies comparable to MPC. If SF sizes are changed from run to run, then resubmission of a design will require redesign. Designs which are small relative to a SF would have relatively long leads between the circuitry and the pads which could affect performance.

4. Summary

The Standard Frame style of IC prototype implementation, which is well matched to electron-beam mask making, makes possible wafer probing and simplifies packaging. Using data from two MPC runs, it has been shown that mask and wafer utilization are about the same for the MPC and SF styles.

C. MOSIS LASER-FORMED CONNECTION EXPERIMENT

Redundancy is necessary in very large (wafer-scale) integrated circuits since processing is imperfect and a certain percentage of the circuitry will not be functional. One approach to the defect avoidance problem is to partition the total circuit into pieces which can be individually tested after fabrication. These pieces are then interconnected using an X-Y grid of conductors. Primary interest centers on the device placed at the vertical-horizontal crossings in the grid. Although a wide variety of possibilities is available, laser zappable links look very promising and have many desirable characteristics such as low "on" resistance, high "off" resistance, visually checkable connections, high reliability, and inexpensive programming equipment.

The DARPA MOSIS process provides only single-level metal, whereas the zappable links built at Lincoln Laboratory use metal-metal structures.[8] Data gathered at Lincoln indicated that there were failure modes involving metal-poly and metal-substrate shorts in various test structures. Thus, the possibility was present that these mechanisms could be exploited to form useful links in the MOSIS context.

A test chip was designed for MOSIS fabrication comprising nine 3-by-3 test arrays of metal-poly links and a similar number of metal-diffusion links. The dimensions of the test links were very nearly the same as those used in conjunction with the Lincoln bulk CMOS facility, making the same probe card and test equipment usable. Many copies of the chip have been fabricated and received at Lincoln Laboratory.

The first attempt at zapping the links looked good visually. Further, it appeared that three times the amount of beam intensity was required to cut through poly as through metal, indicating that it would be possible to make a metal-poly link without shorting through to the substrate. Difficulties with the test facility have prevented checking the electrical properties of the links, but this is being actively pursued.

Another test chip is being considered for implementation. This will have two other structures which might be used for zappable links in the MOSIS process. Depending on the results of the electrical tests on the current link designs, this will be laid out and submitted for fabrication in the near future.

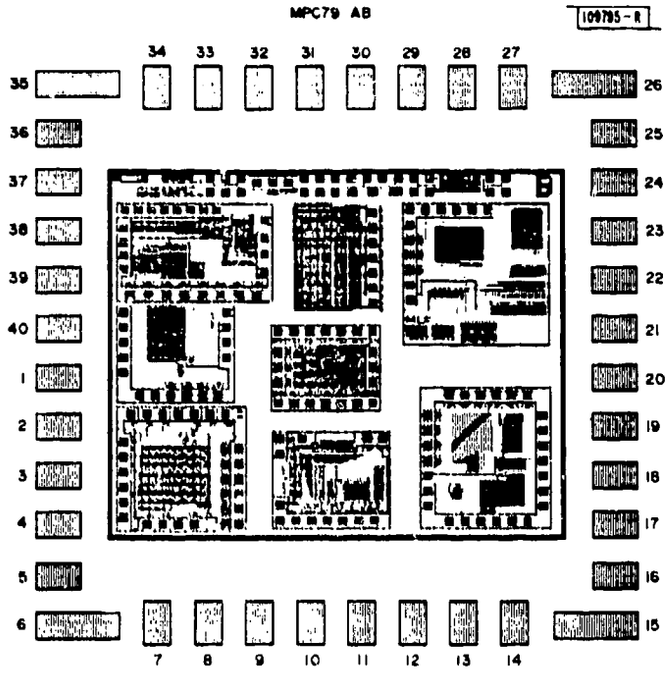


Fig. II-1. Example of a die from MPC79.

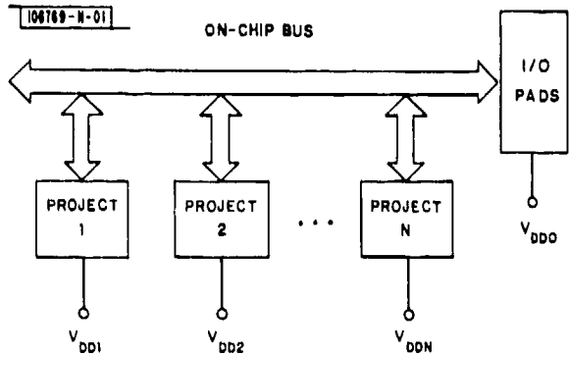


Fig. II-2. Schematic representation of dynamic bonding.

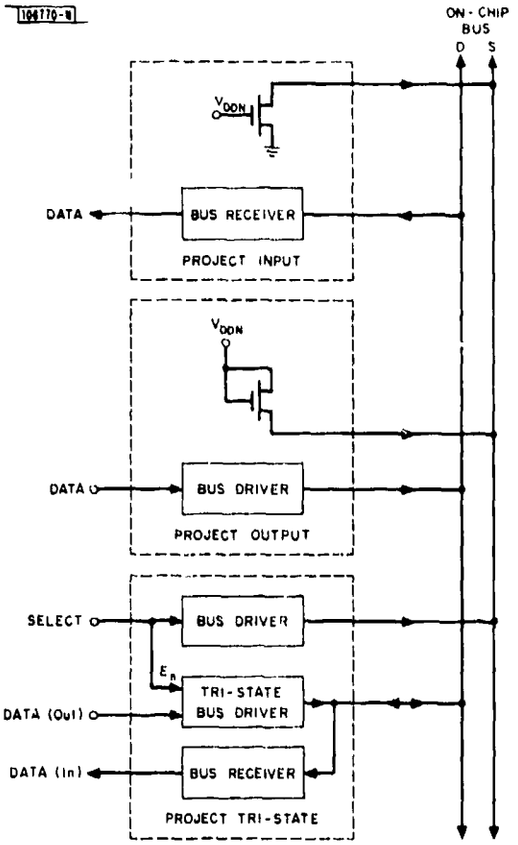


Fig. II-3. Dynamic bonding project I/O circuits.

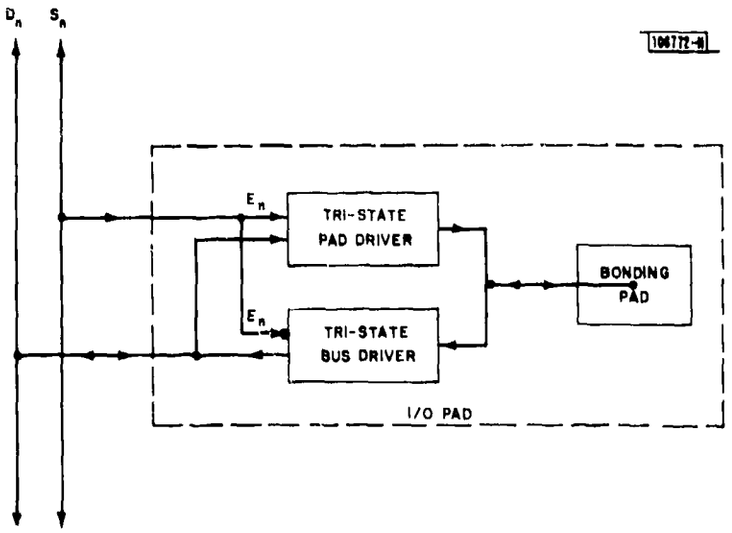


Fig. II-4. Dynamic bonding die pad I/O circuit.

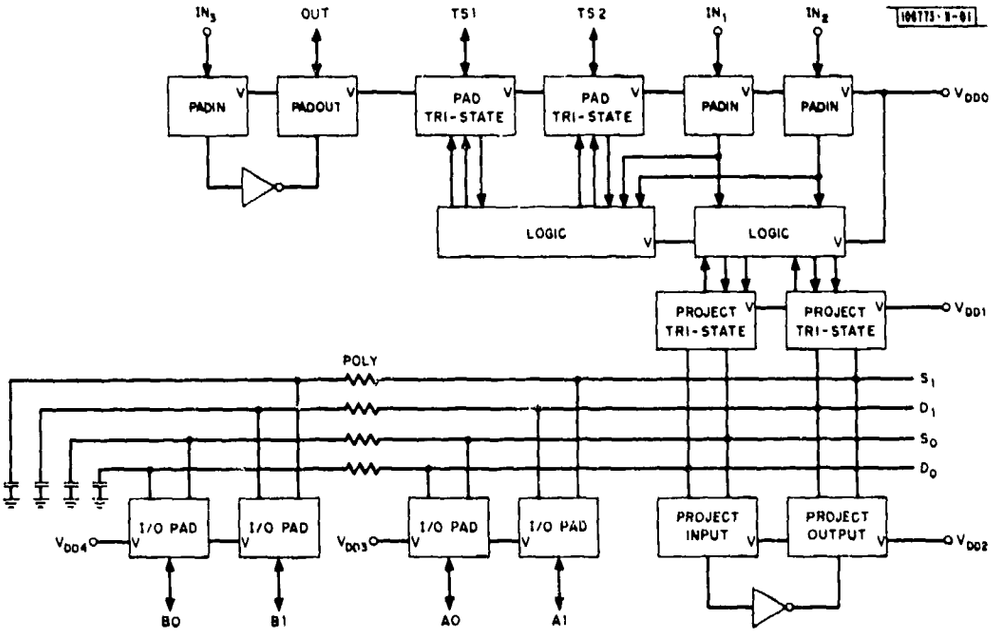


Fig. II-5. Dynamic bonding test project.

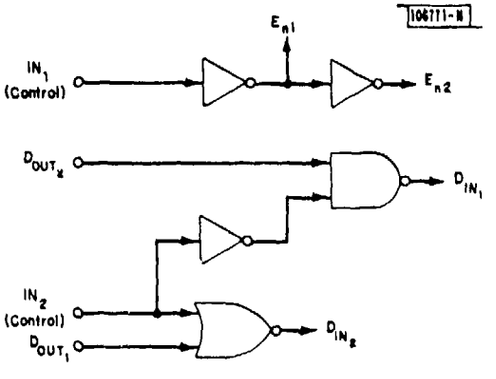


Fig. II-6. Logic circuitry for dynamic bonding test project.

III. DESIGN AIDS FOR RVLSI

A. INTRACELLULAR

1. LICL – Lincoln Integrated Circuit Language

LICL is a simple layout language which allows chip designers to describe IC masks in a high-level text form.[9] Representing the chip as a program rather than graphically allows the designer to implement arbitrarily complex routing strategies and conversions from, say, logic descriptions into canonical logic forms such as gate matrices, gate arrays, and programmed logic arrays. The output of LICL programs is CIF 2.0 suitable for analysis and simulation with tools developed at the M.I.T. Laboratory for Computer Science. The CIF output is compatible with the MOSIS fabrication service.

LICL "programs" are actually written in "C," the primary programming language used on UNIX(TM) systems. (LICL development was done using a PDP-11/70 computer, and LICL has been adapted for VAX UNIX since the Lincoln machine has been installed.) Embedding LICL in C rather than writing a separate compiler allowed quick implementation, but also created a number of problems. Mainly, there is an extreme decoupling of the program which actually generates the CIF output and the source code that describes this process. Thus, when an error is encountered such as a "box with zero width," no pointer back into the source code is available to indicate where the problem lies. In spite of the drawbacks, other ARPA VLSI research groups chose the same approach: the M.I.T. Artificial Intelligence Laboratory uses LISP, California Institute of Technology and USC Information Sciences Institute use SIMULA, Bolt Beranek and Newman's SSL is a distant relative of LICL and is embedded in BCPL, and Stanford University's CLL is also in C.

LICL programs deal with two "normal" data types and two extended types. The normal types are integers, typically used to specify locations on a one-lambda grid, and character strings which are needed to specify and subsequently look up names in a data base associated with a design. The extended data types are POINTs which are simply X-coordinate, Y-coordinate, Layer triples, and ITEMs which are pointers to the LICL structure for (a partial) chip design. Given X, Y, and Layer, a POINT variable Pt can be set by Pt = Point(X, Y, Layer). Conversely, a POINT can be decomposed by statements such as ptX = Pt → X, which extracts the X component of the POINT structure Pt.

The simplest ITEM is a rectangle of some material. The Rect(Layer, XMin, YMin, XMax, YMax) function returns an ITEM which is the representation of the required rectangle in the data base. A different and sometimes more convenient way of describing a rectangle is Box(Layer, Length, Width, XCenter, YCenter) which is closer to the CIF "Box" primitive.

Wire(N, Layer, Gage, X1,Y1, X2,Y2,...,XN,YN) returns an ITEM which is a wire of width "Gage" and having centerline points X1,Y1, X2,Y2, through XN,YN. Sometimes it is more convenient to describe a wire by starting with the absolute coordinates of an end point and then using incremental displacements for the others: DWire(N, Layer, Gage, X1,Y1, dX2,dY2,...,dXN,dYN).

In order to combine rectangles, wires, and other ITEMs, the function Merge(N, I1, I2,...,IN) is provided. Also, there are the following geometric operators which apply to ITEMs: MirrorX(I) MirrorY(I), RotCCW(I), RotCW(I), Move(I, dX,dY), Home(I), and Align(I, Name, Point). Home is a special case of Move which forces the upper-left corner of its argument to be at the origin, a fairly standard convention. Align moves its first argument so that the named point in that ITEM coincides with the third argument.

In addition to forming structures, one may ask questions about them. The following functions can be used to allow programs to do placement of substructures: Top(I), Bottom(I), Left(I), Right(I), Width(I), and Length(I).

The Cell(I, "Name") function declares that the argument ITEM is a cell with name "Name." Cells are special kinds of ITEMs that can have names declared in them that represent connection points. For instance, we might write,

```
Adder = Cell(Ad0, "XAddr");
Terminal(Adder, Point(10, 20, NP), "In");
Terminal(Adder, Point(10, 40, ND), "Out");
```

This will define the two names XAddr.In and XAddr.Out. After the Adder cell has been translated, rotated, etc., the input point can be found by

```
AdIn = Find(f(Adder), "XAddr.In");
```

which defines the POINT named AdIn. Cells can be used as parts of other cells, and the names concatenate in the normal left-to-right way. For example, if Adder is used in a bigger cell called CPU, the adder input name would be CPU.XAddr.In.

The function Repeat(item, Nx,Ny, Dx,Dy) returns an ITEM which is a LICL array of Nx * Ny copies of "item." Repeat calls the Cell function internally to give integer names of the form X.Y to each instance of "item." For example,

```
Repeat(Adder, 2,3, Length(Adder), -Width(Adder));
```

forms a 6-element array. The input terminals have the following nan .

0.0.In	Upper-left corner
1.0.In	Upper-right corner
0.1.In	Middle row, left
1.1.In	Middle row, right
0.1.In	Bottom-left corner
1.1.In	Bottom-right corner

Frequently, the PDP-11/70 is too small to hold the data base for a typical chip. This is due primarily to the rather limited address space that a process sees. To cope with this limit, two pseudoitems have been added to LICL – Include("cif-file") and Call(Symbol-number). These are useful when composing a design from cells supplied from elsewhere in the form of a CIF file (e.g., the Stanford cell library).

There are several libraries and extensions of LICL available. Pads, bits, and bfrs are the various parts of the original Xerox PARC library. CR is a very simple channel router which assists the chip designer in making connections automatically. PF is an experimental feature which provides users with only four different chip sizes (full, half, quarter, and eighth die) and standard positions for the bonding pads (cf. Sec. II-B). PF will enable wafer probe testing before packaging and more efficient utilization of the wafer area. Other extensions such as a register array generator are being investigated at the present time. The PLA generator extension is a candidate for implementing some of the new folding algorithms which have been reported recently.

To date, LICL and its extensions have been used successfully to generate three project chips for MOSIS.

2. GAMMA – Design Automation for the Dense Gate Matrix Discipline

An integrated circuit or logical cell design may be specified at several different levels of abstraction. The lowest level is the actual geometric mask layout. This level contains the most information and is consequently the most difficult and time consuming to specify. Above this level, an abstraction known as sticks[5] is often used, which can specify the topology of a circuit without the actual geometries of the design rules. By relieving the designer of the necessity to concern himself with geometry, the design task becomes significantly easier. The sticks design, however, must eventually be transformed to a full geometric mask specification. Above the sticks level, a circuit may be specified as a schematic or a network of transistors. Since logic gates can be treated as small transistor network macros, the design task becomes similar to conventional logic design. At even higher levels, a circuit can be specified either by the logical function it performs or the algorithm it implements. As one progresses to higher levels of abstraction, the design task becomes simpler as less detailed information must be presented. Still, these high-level descriptions must be converted somehow to the low-level mask design for fabrication.

Others have done work in automating the process of converting some of the different high-level abstract design formats down to low-level mask data. Programs such as Cabbage have been written which attempt to convert from sticks to geometric mask layouts. Such programs accept arbitrary stick diagrams, flesh them out naively according to the design rules, and then compact the resulting design. Similarly, elsewhere in the DARPA community attempts are being made to construct mask layouts from functional specifications of a circuit in the form of a LISP program. The work described here concerns itself with building masks from the transistor network level of abstraction. We have chosen to concentrate on this intermediate level of abstraction because we feel that the functional level is too ambitious while the sticks level still requires significant design effort.

As part of the RVLSI project, a language known as HISDL (Hierarchical and Iterative Structure Description Language) was developed[4] which is well suited for specifying interconnections between components. In our case the lowest-level components can be transistors, which allows us to use HISDL to specify a circuit as a network of transistors. The hierarchical nature of HISDL is particularly useful, as gates may be defined in terms of transistors and larger functional blocks in terms of gates which greatly simplifies the design process. An example of a hierarchical HISDL description of a D Master Slave Flip-Flop is shown in Fig. III-1. The goal of this project is to convert the logical net list (LNET) output of the HISDL program into complete mask specifications.

Normal mask layouts have many degrees of freedom. So many different decisions must be made when laying out a circuit that it is inconceivable to automate the process of generating an optimum arbitrary mask. By constraining the domain of allowable design formats, however, the number of decisions required is considerably reduced. This often results in layouts which take more area than comparable arbitrary ones. This is the case with such formats as conventional gate arrays. A trade-off thus exists between the potential for automation vs the resulting efficiency of design. It is therefore desirable to develop a format which constrains the design enough to make it manageable, yet still leads to designs which are only somewhat less efficient than handcrafted ones.

In a recent paper by Lopez and Law,[10] a layout discipline was proposed for CMOS logic which seems to meet the above criteria. This discipline is known as the "dense gate matrix" layout method. Briefly stated, designs using this discipline have vertical polysilicon lines acting as both I/O lines and as common gates for transistors. The diffusion lines run horizontally crossing the poly lines, thus creating transistors. Metal lines also run horizontally connecting transistors. This method differs from conventional gate arrays in two ways: First, the transistors are not in fixed locations; second, all mask layers change from circuit to circuit, not just the metal mask. Lopez and Law claim that designs done by hand using this discipline compare favorably with arbitrary hand designs of the same circuit. No mention is made, however, of any algorithm which implements this design discipline. A concise yet detailed description of the gate matrix discipline appears in Ref. 11.

When developing an algorithm for implementing the design discipline, it is necessary to first accurately specify the discipline. We felt at first that the discipline expressed in Ref. 11 was too general to specify concisely. Therefore, we chose to restrict this discipline even further by allowing only straight horizontal diffusion paths without the vertical stubs or "T"-shaped structures that appear in the original discipline. This restriction precludes any possibility of overlapping metal and diffusion, since diffusion paths must terminate at both ends with a contact cut, thereby shorting out any metal path running over diffusion.

With this restricted discipline, it is now possible to enumerate all possible layout situations. A detailed analysis of the allowable interconnections in CMOS logic design shows that, to specify the transistor network, three connection types suffice, namely: common gate-to-source, drain-to-source, and drain-to-common gate. By definition, we assign the source of a transistor to be the left edge of a diffusion path and the drain to be the right edge. This results in two subcases for each of the above three connection types, depending upon the ordering of the common gates involved. A third subcase arises when there are intervening polysilicon lines between the two common gates being considered which requires a metal crossover to avoid conflict with the poly lines between them. All nine cases are summarized in Fig. III-2.

The algorithm implemented consists of two major phases. The first phase chooses a "good" ordering of the vertical polysilicon lines. Once this ordering is established, phase two places the transistors on their appropriate common gate and connects their sources and drains according to the nine cases enumerated above.

A heuristic is used to find a good ordering for the common gates. As can be seen from Fig. III-2, any connection such as a drain-to-source connection, which has the common gates of the two transistors in the wrong order, will require extra rows to handle. It is therefore desirable to minimize these feedback connections in any acceptable ordering of the common gates. It can also be seen that, given two transistors which are in series and lie on adjacent common gates, the diffusion path of the transistors may be continuous with no intervening metal. It is presumably desirable to minimize as much as possible the amount of metal used, and therefore require all common gates of transistors which have drain-to-source connections to be as close to each other as possible. Phase two thus consists of three parts. First, a directed graph depicting the dependency of common gate orderings is developed. The nodes of this graph are the common gates. An arc exists between two nodes if there is either a connection from the first gate to a source of a transistor on the second gate, a connection from the drain of a transistor on the first gate to the second gate, or a connection from the drain of a transistor on the first gate to the source of a transistor on the second gate. Therefore, this arc means that the second

common gate must come after the first in order to avoid a feedback connection. This graph is actually weighted to reflect the number of dependencies each single arc represents.

If this graph is acyclic, then there exists some ordering of the nodes which will not require any feedback connections; but, in general, this is not the case. Phase one therefore attempts to find a set of arcs with minimum total cost which, when removed, will yield an acyclic graph. This is an NP-complete problem requiring that an efficient heuristic search technique be used. A topological sort is done on the resulting acyclic graph to yield an ordering which has minimal feedback paths. Of all the valid orderings, the topological sort chooses one which tends to minimize the amount of metal and metal contacts used. It does so by computing a cost function of an ordering based on the sum of the distances between dependent gates. The cost of an arc between two common gates is zero if they are dependent due to a series transistor and are adjacent, since a continuous diffusion path is used and no metal is required. Otherwise, a large initial penalty is added to the cost for requiring a contact cut, which is added to the distance between the two common gates in the ordering. No efficient algorithm is known for finding an optimal weighted topological sort short of enumeration. Therefore, an initial solution is found constructively which is improved by pair-wise swapping. Results show, however, that the constructive initial sorter, which does some alternative checking, performs well enough alone without the pair-wise swapping.

Phase one of this algorithm was written and debugged. This program is known as "GAMMA," from Gate Matrix. Using the ordering suggested by GAMMA, several layouts were done by performing the case analysis of phase two by hand. Results of the design of a SR Flip-Flop and a D Master Slave Flip-Flop are shown in Figs. III-3 and III-4. When comparing the design of the D Flip-Flop with the design in Fig. III-5 which is copied from Ref. 11, it is clear that the automated design is inefficient. After analyzing the results it seems that the source of inefficiency lies not in the heuristic used to order the common gate lines, but in the restriction against overlapping metal and diffusion. The "T"-type structures and their generalizations, which we call clusters, are heavily used in Ref. 11 for implementing the parallel and series transistor networks of NAND and NOR gates. By reversing the left-right orientation of transistors, and having a diffusion stub act as the common drain of two transistors, a single metal path may run over the diffusion to connect all the sources together. If several clusters share the same metal path, several rows can be compacted into one. A pathological example of this inefficiency is shown in Fig. III-6(a-c). It is therefore clear that, in order to produce efficient designs, the discipline must be expanded to include clusters.

A preliminary case analysis has been done on the cluster discipline. These cases have not yet been fully formalized, as they are more detailed and comprehensive than the cases shown in Fig. III-2, and hence will not be given here. Using these cases however, the SR Flip-Flop and D Flip-Flop layouts have been redone as shown in Figs. III-7 and III-8. These show great improvement over the previous designs, and approach the efficiency of the hand-done layout in Ref. 11. Work is continuing toward the improvement of the cluster idea and its eventual implementation.

3. Cell Design Approach Based on Regular Structures

A methodology has been developed for the implementation of cells which can be described as combinations of finite state machines and register transfers. A software language has been

defined to implement this methodology, called MACPITTS,† having a syntax similar to LISP but very different semantics. The fundamental difference between MACPITTS and conventional software languages is the parallelism it encourages which is particularly well suited to describing signal-processing-oriented structures.

As advances in VLSI technology allow larger and denser ICs to be fabricated, designers are attempting the implementation of more ambitious systems. The design of such systems has become exceedingly complex, causing an increased interest in design automation tools. These tools accept a more concise abstraction of the design than the mask layout as input, and produce a detailed mask layout as output.

Many design aids are geometric, dealing with the placement and layout of boxes, wires, mask levels, etc. LICL is a good example of such a design aid (cf. Sec. A-1 above). A second-level design aid accepts a topological or schematic description of a design and helps the user produce an IC mask. An example of this type is the dense gate matrix discipline (cf. Sec. A-2). The advantage of the geometrical and topological aids is that all useful logic structures can be so-described. This is convenient for small, random-logic designs, but it cannot easily manage large regular structures such as RAMs, ROMs, and PLAs. Specifying the contents of a ROM by its circuit description would not be the optimum use of a designer's time.

A third type of CAD system accepts functional descriptions. MACPITTS is a CAD system of this type. We believe that as designs become more complex, the functional description becomes more concise and suitable, and is ultimately always necessary.

Many tasks involve the manipulation of data stored in registers and control sequences and signals. Such tasks may be concisely specified by a combination of register transfer statements and a finite state machine for control. A MACPITTS program is just such a description. Conversion of such programs into hardware is called "compilation." Despite the analogy with software compilation, this does not imply the compilation of programs into code which executes on a general-purpose machine. It should be clearly understood that MACPITTS automatically constructs the exact hardware resources for the compiled task.

An important concern is that the compiled design should have performance comparable to designs generated by other means. Many signal-processing tasks are naturally implemented as designs which make use of large amounts of parallelism. This methodology must be capable of generating such designs. Conventional microprogramming languages are usually sequential in nature and thus are not suitable for specifying simultaneous action. Two solutions exist. One alternative is the automatic detection of parallelism implicit in a sequential algorithm and the construction of an equivalent parallel program. The other alternative is to define a language with explicit syntactic and semantic constructs for specifying parallelism. We have chosen the latter method, since we feel that the former is a very difficult problem requiring a prohibitive investment in time and effort. The method of including parallel constructs in the language was motivated by Hoare's work on Communicating Sequential Processes[12] and bears resemblance to CSPs. We envision, however, a future front end for our system which will perform the parallelism detection. The support for parallelism in the system has been an important factor in the choice of language constructs and target architectures.

A design will consist of a small number of different modules of several types. These modules will be interconnected using placement and routing routines which are in the process of

† The name "MACPITTS" is a tribute to the early contributors to the field of finite state automata: McCulloch and Pitts.

development. Optimality of a solution is not of primary concern since the ratio of interconnect requirement to module size is very low. Rather, the placement and routing algorithm should require a minimal amount of human assistance and guarantee a solution as often as possible. This does not seem difficult since only a small number of modules need be interconnected, implying that even exponential time algorithms may suffice.

Each individual module will be generated by the appropriate routine based on its type. Two major module types have been identified at present, namely finite state machines and register arrays. Future module types may include RAMs, ROMs, and multipliers. A general facility is contemplated for merging modules from other sources, perhaps even from a dense gate matrix-like circuit description, and interfacing them with the standard module types. Finite state machines (FSMs) are implemented as PLAs with clocked feedback. They are used to implement the control structure which sequences register array operations. FSMs communicate with other modules and with the outside world through signals. The source language for MACPITTS allows specification of a sequence of output signals to be asserted conditionally dependent on the status of input signals. The semantics of those signals which connect to modules other than other FSMs are not formally specified in the source program at present. Future versions of MACPITTS will include constructs to specify explicit register transfer operations and generate the required signals to effect those actions.

Parts o MACPITTS will be useful even before a fully embellished system is completed. Specifically, the register array and FSM compiler could be used in the context of the LICL system. A version of the register array generator has already been incorporated into LICL. The FSM compiler is not quite finished, but no conceptual problems appear to remain. An FSM simulator is presently functional. Future additions to MACPITTS will include extensions to CMOS and enhancement of cell testability and fault tolerance through automatic test vector generation, built-in diagnostic modes, and hardware augmentations for error detection/correction.

B. INTERCELLULAR

1. Placement, Assignment, and Linking Automation

a. Introduction

The design of RVLSI systems can be divided into two distinct stages. The first stage involves the specification of the logical system and the design of the physical resources to meet that specification. The design of those resources can be further decomposed into the cell design, placement of those cells, and the definition of the interconnect. This stage is necessary before the fabrication of the wafer. At this point, designers consider such issues as optimum cell size, amount of cell redundancy required, and connectivity channel capacity. The answers to these questions can be very process-technology and design dependent.

The second stage comes after fabrication, at which point the system design is instantiated on the available physical resources. This stage likewise divides into three phases: test, assignment, and link. The test of the wafer finds nondefective cells and interconnect. The assignment phase maps the required logical elements onto the available, nondefective cells. The link phase then finds paths through the functional interconnect that instantiate the logical nets. The assignment and linking phases of this stage were the first to be studied and automated.

Automation for performing these tasks is known collectively as PAL, an acronym for placement, assignment, and linking. The preliminary version evolved out of the separate tools that

were being developed to explore problems that are unique to RVLSI as a research system. It was used to study the second stage of RVLSI design, i.e., the various aspects of assigning and linking a whole-wafer system. As such, it was never intended for actual production purposes, although certain algorithmic techniques learned during its development might be. Once a specific target application, such as the packet radio integrator (cf. Secs. IV-B-1, -2), was identified, parts of the PAL package were also used to model wafer yields.

The preliminary PAL system was designed to be as general and as technology-independent as possible. At that time a link technology had not been identified. Therefore, parts of the system whose algorithms depended on link "costs" were parameterized. In that way, the impact of different link technologies on linking algorithms could be measured. However, certain interconnect technology assumptions were made. The interconnect was to be layed out on a rectangular (Manhattan) grid. Also, the wires of the interconnect (or segments) were assumed to be fixed, implying that the placement, orientation, and length of each segment was defined at fabrication time. Finally, whatever the technology, a link was assumed to be bidirectional such that a signal could propagate through from either side. These assumptions deeply affected the design of critical parts of the PAL system. For example, the Manhattan grid assumption prompted the development of a unique placement language, PLATEXT, specifically tailored to the definition and placement of cells, channels, and link blocks on a Manhattan grid.

Figure III-9 depicts the structure of the original PAL system. There are three types of information needed to assign and link a wafer: the logical description of the system, the physical wafer description, and a map of good cells deriving from the test results. Certain programs or languages in PAL provided a means to easily define this information. The specific programs were:

> HISDL – used to transform a textual description of the logical net list. The syntax and semantics of this language were described in detail in Ref. 4. Since then, a preliminary version of the program has been written and used extensively.
>
> PLATEXT – this program transforms a textual description of the physical layout of the wafer into a fully expanded wafer specification.[13]
>
> Calma Graphic Interface – provides same information as PLATEXT, but the wafer design is done on the Calma machine.
>
> Probe – models test results by giving a list of bad cells, given the size of the two-dimensional array of cells and cell defect probabilities.

Experiments were conducted within the PAL system using many assigners and linkers. Since the system was designed with modularity in mind, with programs communicating through human readable interface files, it was easy to substitute new assigners or linkers to test new ideas. The preliminary linker results were reported in Ref. 4. The latest assigners and related results are described below.

b. Assignment

Assignment is the process of binding logical to physical cells on the tested RVLSI large-area chip. It is convenient to classify assigners as constructive or nonconstructive. A constructive assignment is one which is guaranteed to be linkable in the absence of interconnect defects, while

a nonconstructive assignment is not guaranteed linkable. A second characteristic of an assigner is its generality, that is, whether it is applicable to assignment problems in general or only to a specific logical and physical design. A general constructive assigner would necessarily include a linker, for only by doing a linking could the linking be guaranteed, except where the interconnect resources were very generous. A nonconstructive assigner may produce a better assignment by consideration of linkability.

Reported here are results obtained with a nonconstructive assigner for one specific class of system organization, and a constructive assigner for the packet radio integrator.

c. Nonconstructive Assigners

Nonconstructive assigners were written for logical two-dimensional arrays of cells with nearest-neighbor connectivity; that is, each cell has one connection to each of its four neighbors. The experiments consisted of generating an 8 × 8 array with some interconnect of fixed segmentation, simulating ten defective wafers by randomly distributing 32 good cells on each array, performing an assignment, and then attempting to link with the graph reduction linker of Ref. 4. The assignment was considered successful if a linking could be performed.

A commonly used placement method in PCB and IC layout is force-directed exchange. An initial placement (assignment) is made, and the force on each cell is calculated based on some measure of its relationship to other cells. Then, some pair of cells is selected for mutual exchange so that some global measure of force is based strictly on physical distance. The three selection methods tried were:

(1) Steepest Descent – the iterations continue until no exchange will reduce total force. This strategy does not guarantee a global minimum.

(2) First Descent – this is similar to steepest descent, except that the first exchange that reduces force in an iteration is implemented.

(3) Force Directed – here the cell with largest force on it is chosen for exchange with another cell in the direction of the force vector.

Three force measures were used:

(1) $dx + dy$ – sum of grid distances.

(2) $dx^2 + dy^2$ – sum of squares of grid distances.

(3) Corrected – a grid distance measurement corrected for the effect of the positioning of the connection point on a cell face.

When the interconnect provided four tracks in each channel, then any combination of strategy and force measure gave a linkable assignment. When the interconnect was reduced to three tracks per channel, some experiments were necessary to find the best segmentation. The dx + dy force was found to be inferior to the other two. The results of more successful experiments were as follows:

Selection	Force	Number Linkable	Execution Time	Comments
First Descent	$dx^2 + dy^2$	1/8	15 min.	–
Steepest Descent	$dx^2 + dy^2$	2/8	14 min.	–

Selection	Force	Number Linkable	Execution Time	Comments
Force Directed	$dx^2 + dy^2$	4/10	20 s	Line of cells
Force Directed	Corrected	5/10	30 s	Logical cells in distortion direction
Force Directed	Corrected	2/10	20 s	Physical cells in distortion direction

Of the ten wafers, eight were assigned and linked at least once. The most fruitful way to extend this work would be to measure force in a way more representative of the actual interconnect. However, the following hashing technique was more successful.

The strategy of the hashing assigner is to iteratively assign logical to physical cells based on a preferred location for each logical cell and trajectories of secondary locations in case of clashes at preferred locations. (The name is by analogy to storage allocation by hashing.) An iteration begins with the most constrained logical cell, i.e., the cell with the most connections to other cells which have already been assigned. This is only one of several possible methods which seem reasonable for choosing the next cell to iterate upon. Next, a trajectory is formulated which orders possible cell assignments in order of least distortion force from the constraining cells. This trajectory includes all locations containing cells of the proper type, including those already assigned. If the most desirable location is currently unassigned, then the logical cell is assigned to it, and the loop is repeated for the next logical cell. If the desirable location is currently unassigned, then the logical cell is assigned to it, and the loop is repeated for the next logical cell. If the desirable location is assigned, then one or two things must be done: either the current logical cell must be assigned to the next location in its trajectory (or beyond), or the logical cell currently assigned to the clash site must be reassigned on its trajectory ("backtracking"). By calculating the difference in distortion force for the current logical cell in its most desirable location vs its next most desirable location, and the difference for the clash cell between its current location and its next backtrack location, it is determined which of the two operations to attempt.

In backtracking, the clash cell's next location may also generate a clash, in which case backtracking proceeds recursively (generating a backtrack "bubble") until a decision is reached as to whether backtracking is worthwhile or not. Alternatively, the next item in the current logical cell's trajectory may clash also. In this case, it is the distortion force of the third location vs the first two which limits the backtrack. This means that the backtrack bubble of the first clash can now be expanded, and a backtrack from the second clash can be considered.

There are some theoretical difficulties here. When a new cell is assigned, those cells already assigned may have new constraints imposed on them, that is, their force gradients will change. This means that the trajectories by which these previous cells were assigned are no longer in the proper order, and the present assignment of these cells may no longer provide a minimum force. The present program takes this into account, to a limited degree, by recalculating the trajectories of the clash cells at every backtrack. This is not completely correct, but gives a good first-order approximation of all the forces acting on the cells.

This minimization method, in conjunction with the corrected $dx^2 + dy^2$ force function, yielded linkable assignments for 8 of 10 wafers in 40 s of computer time under the same conditions as reported above.

d. Constructive Assignment for the Integrator and Redundancy Studies

The packet radio integrator which is being built as a first demonstration of RVLSI is a logical array of 64 cells in eight columns of eight (see Sec. IV-B-2). There are two types of logical connections: broadcast to every cell, and cell-to-cell within the columns. The broadcast connection does not affect assignment strategy in any way, so only the one-dimensional nearest-neighbor connections need to be considered. For each assumed interconnect capability, the assignments were constrained so as to guarantee linkability. The array of 8 × 8 cells is mapped onto a physical array of x rows by y columns, or, for generality, m rows by n columns are mapped onto x rows by y columns.

The array yield for several of the strategies can be determined analytically making use of a simple formula. Given a set of R cells, each with an independent probability P of functioning correctly, then the probability of at least S of these cells functioning correctly is given by:

$$\text{YIELD}(P,S,R) = \text{SUM}\ \{I,S,R\ [(RII) * P ** I] * [(1 - P) ** (N - I)]\} \quad .$$

An upper bound on array yield is determined by assuming an interconnect which allows any arbitrary assignment of cells; this is termed unconstrained assignment. The yield is YIELD(P,M * N,X * Y). Figures III-10 and III-11 show yield for an 8 × 8 array for x,y = 8 to 15 for cell yields of 50 and 70 percent, respectively. The top number for each x,y pair is the unconstrained yield.

The first attempt at a constrained analytical assignment chose to map complete logical columns onto physical columns. A die designed for a M * N logical array would have physical dimensions M * Y. Of these y physical columns, at least N are required to contain no defective cells. The probability of a column being perfect is YIELD(P ** M,N,Y). This scheme has as its advantage the requirement for a minimally restructurable interconnect. Physical columns are not restructured internally. Only the connections between the columns and the outside world need be permuted.

Instead of providing redundant columns as done previously, it is possible to provide redundant rows. Such a scheme would require interconnect which would allow skipping of defective cells in a physical column. A die designed for an M * N logical array would have physical dimensions X * N. The probability of a column having at least M good cells is YIELD(P,M,X). The die yield for this model is therefore YIELD [YIELD(P,M,X),N,N] or YIELD(P,M,X) ** N.

The previous two orthogonal strategies may be combined to allow mapping the columns of an M * N logical array on an X * Y physical die with the constraint that physical interconnectivity is restricted to remain within columns. The yield for this model is given by YIELD [YIELD(P,M,X),N,Y]. This reduces to the previous case when Y = N, and to the first case when X = M. This strategy is known as "unlimited skip" from the required capability for skipping over an arbitrary long sequence of defective cells. This is a reasonable capability to provide for when assuming arbitrary segmentation. Only one track per channel is needed for each daisy chain line, as it may be segmented between every point where the signal enters and leaves a good cell. When a cell is defective, the cell is disconnected from the track and the track is left unbroken to allow the signal to skip over the cell.

On the other hand, unlimited skip is expensive to provide for under a fixed segmentation assumption. In actuality, unlimited skip requires a maximum skip of X – M cells. The minimum segment length needed to skip K consecutive cells has length K + 1. When skewing segments of this length, K + 1 tracks are required to allow for all possible conditions. As the required channel width increases with the skip capability provided, it is desirable to determine

the effect of limiting the skip capability. The limited skip strategy is called SKIP(K). Both the assignment algorithm and the routing template are parameterized with the maximum skip value K.

The original formulation of the SKIP(K) constraint required not only that the cell-to-cell separation not be greater than K, but, additionally, that the two end cells not be further than K cells from the edges. Though intuitively sound, this constraint is too harsh for large redundancy. For example, when K = 1 and M = 8, for any X larger than 17 the yield is zero as there is no way to span the column with only 8 cells even though the logical design requires exactly 8 cells. A milder formulation of the SKIP(K) constraint removes the edge criteria. End cells, which are too far from the edge to be using one segment, can be connected by chaining several free segments together. A recursive function used to calculate the yield of this model is given below:

```
y = lambda(p,m,n,k)
  (y(p,m,n,k,0,1)
  where rec y = lambda(p,m,n,k,i,f)
    (if m > n
      then 0
      elif m = 0
      then 1
      elif f
      then p * y(p,m-1,n-1,k,0,0) +
           (1-p) * y(p,m,n-1,k,0,1)
      elif i = k
      then p * y(p,m-1,n-1,k,0,0)
      else p * y(p,m-1,n-1,k,0,0) +
           (1-p) * y(p,m,n-1,k,i+1,0) fi ))
```

In Figs. III-10 and III-11, the second number is array yield for unconstrained skip and the third is for SKIP(1) for 8 × 8 logical arrays. At 100-percent redundancy, unconstrained skip gives good yield with long columns for 70 percent but not for 50-percent cell yield, while SKIP(1) gives very poor array yield for cell yields of 50 and 70 percent.

For 50-percent cell yields, a less constraining interconnect must be used. One alternative scheme would map each logical column onto a preferred physical column, but not require all cells to be located in that single column. Instead, surplus cells from up to L columns to each side of the preferred column can be included in the assignment for that logical column. This new constraint is referred to as MIGRATE(L) and is used in conjunction with the SKIP(K) constraint. An extension of this strategy allows a logical column to move left or right constrained only by how far the end cell can reach. This bidirectional skipping is termed BISKIP(K,L). Figure III-12(a-b) shows that six cells can be reached with BISKIP(1,1) and fourteen with BISKIP(1,2). The number of horizontal tracks is 2L + 2, and the number of vertical tracks is 2K + 2.

An assigner for this strategy was coded. It uses an essentially exhaustive combinatorial search. By ignoring some less likely possibilities in the search space, the algorithm returns both positive and negative results quickly. With 2L + 2 horizontal tracks and 2K + 2 vertical tracks, no linking failures have been observed, but we have not proven that it is guaranteed. The last result in Figs. III-10 and III-11 is yield for BISKIP(1,2) averaged over ten Monte-Carlo simulations. For 50-percent cell yield and x = y = 12, the yield for BISKIP(1,1) is 36 percent. Figure III-13 is a plot of yield as a function of y for x = 14 at 50-percent cell yield for the various strategies.

These assignment experiments dramatically illustrate, for this particular organization, the yield loss from interconnect constraints which are too restrictive. It should be emphasized that the use of segmentable (laser zappable) interconnect may make the less-constrained assignments less costly in interconnect space. Assignment and linking with segmentable interconnect are now being developed for the integrator, with the circuit design considerations factored in.

e. Summary

While the preliminary PAL system is now obsolete, it must be emphasized that much was learned from it. Several designs passed through it, providing statistics on wafer yield and channel capacity. Useful techniques for assignment and linking were invented and, generally, the designers developed a reasonable feel for how difficult or easy certain tasks might be. All this forms a basis for the improved, production-oriented PAL system which is described below.

2. Production-Oriented Assignment/Linking System

The current PAL system is designed for implementing real projects in a real whole-wafer technology. This is in contrast to the original system, which was designed to conduct simulated experiments. Specifically, the current form of PAL will be used in the production of the phase 0 and phase 1 integrator wafer (see Secs. IV-J -1, -2).

The most significant change from the original PAL system is the inclusion of arbitrary segmentation.† Arbitrary segmentation means that interconnect conductors can be cut at any point, forming two electrically independent conductors. If this segmentation facility is used effectively, it can considerably reduce the interconnect required for redundancy.

Also included in the present PAL system is a full hierarchy of commands predicated on the laser zapping and link technology. At the lowest level, these commands move the x-y positioning table and trigger the laser. Commands at the next level deal with the link and conductor entities on the wafer: connecting, disconnecting, and segmenting. The final level of command is nominally the human interface. It is here that linking and assigning intelligence are provided. PAL is thus potentially conceptually interactive, the only current bottleneck being at the laser-zapping hardware where interactive probing/zapping presents some mechanical difficulties that are not yet resolved.

C. DESIGN RULE CHECKING AND CALMA-CIF INTERFACE

With current IC complexity, it is both impossible and undesirable to ensure integrity of layout via visual inspection of masks and checkplots. Furthermore, detecting errors by iterating

† Arbitrary segmentation is a consequence of the laser-zapping link technology. As such, it was not considered in the original PAL system which was predicated on electronic linking methods, such as MNOS.

through fabrication is both expensive and very slow. And, finally, there is a class of nonfatal errors which may remain undetected and may decrease yield. Thus, a rules checker is an essential tool for IC layout.

However, sophisticated rules checkers are sufficiently complex that, in order for their development and use to be cost-effective, they must meet the following criteria:

(1) Efficiency – large circuits must be checked with finite computer resources.

(2) Low-False-Alarm Rate – it does little good to find a dozen errors and return them to the designer with thousands of false alarms.

(3) Technology Independence – rules must not be "hard-wired." It is not cost-effective to write a new rules checker for every process change.

(4) Good User Interface – to ensure usage of tool.

The MDRC (mask design rule checking) system, developed under a companion Air Force-sponsored program, satisfies the above criteria.[14] The system comprises two modules: a mask-processing machine and a macro translator. The Mask Processing Machine (MPM) executes low-level mask instructions (software evaluation) which are categorized as follows: preprocessing, spacing, logical, topological, and input/output. The macro translator provides a high-level user interface to facilitate the writing of programs for the MPM. This is a two-pass translator, which performs a macro expansion and syntax check. The designer may draw from a predefined library or define his own macros.

Work has begun on coding the MDRC system on the VAX computer. With the addition of instructions to read and write CIF, MDRC (VAX) will provide a very flexible rules checking capability. MDRC, when coupled with a data format standard such as CIF, could be a powerful ARPA network service. A remote user could send a CIF description of his chip, and a CIF description of his errors would be returned.

A MANN-CIF translator has been written and is now operational on the VAX. This utility permits transfer of graphic files from the Calma GDS system to the VAX. Also, a CIF-MANN translator will soon be operational on the VAX. This program will permit transfer of data from the VAX to the Calma, and also provide the capability to produce MANN pattern generator tapes on the VAX.

GAMMA January 9, 1981

```
STRUCTURE DFLIPFLOP(D,S,R,Q,QNOT,PHI,PHINOT,VDD,GROUND)
  IN D,S,R,PHI,PHINOT,VDD,GROUND
  OUT Q,QNOT
  COMPONENTS MASTER,SLAVE : ANNOR5, A,B : NOR2
  BEGIN
    /VDD,MASTER.VDD,SLAVE.VDD,A.VDD,B.VDD/
    /GROUND,MASTER.GROUND,SLAVE.GROUND,A.GROUND,B.GROUND/
    /D,MASTER.IN<1>/
    /PHINOT,MASTER.IN<2>,SLAVE.IN<2>/
    /PHI,MASTER.IN<3>,SLAVE.IN<3>/
    /S,MASTER.IN<5>,SLAVE.IN<5>/
    /R,A.IN<2>,B.IN<2>/
    /MASTER.OUT,A.IN<1>/
    /A.OUT,MASTER.IN<4>,SLAVE.IN<1>/
    /SLAVE.OUT,B.IN<1>,QNOT/
    /B.OUT,SLAVE.IN<4>,Q/
    END
  ENDSTRUCT

STRUCTURE ANNOR5(IN,OUT,VDD,GROUND)
  IN IN<1:5>,VDD,GROUND
  OUT OUT
  COMPONENTS P1,P2,P3,P4,P5 : PCMOS, N1,N2,N3,N4,N5 : NCMOS
  BEGIN
    /VDD,P1.SOURCE,P2.SOURCE/
    /GROUND,N2.DRAIN,N3.DRAIN,N5.DRAIN/
    /IN<1>,P1.GATE,N1.GATE/
    /IN<2>,P2.GATE,N2.GATE/
    /IN<3>,P3.GATE,N3.GATE/
    /IN<4>,P4.GATE,N4.GATE/
    /IN<5>,P5.GATE,N5.GATE/
    /P1.DRAIN,P2.DRAIN,P3.SOURCE,P4.SOURCE/
    /P3.DRAIN,P4.DRAIN,P5.SOURCE/
    /P5.DRAIN,N1.SOURCE,N4.SOURCE,N5.SOURCE,OUT/
    /N1.DRAIN,N2.SOURCE/
    /N4.DRAIN,N3.SOURCE/
    END
  ENDSTRUCT

STRUCTURE NOR2(IN,OUT,VDD,GROUND)
  IN IN<1:2>,VDD,GROUND
  OUT OUT
  COMPONENTS AP,BP : PCMOS, AN,BN : NCMOS
  BEGIN
    /VDD,AP.SOURCE/
    /AP.DRAIN,BP.SOURCE/
    /BP.DRAIN,OUT,AN.SOURCE,BN.SOURCE/
    /AN.DRAIN,BN.DRAIN,GROUND/
    /IN<1>,AP.GATE,AN.GATE/
    /IN<2>,BP.GATE,BN.GATE/
    END
  ENDSTRUCT

CELL PCMOS(GATE,SOURCE,DRAIN)
  IN GATE
  INOUT SOURCE,DRAIN
* ENDCELL

CELL NCMOS(GATE,SOURCE,DRAIN)
  IN GATE
  INOUT SOURCE,DRAIN
  ENDCELL
```

Fig. III-1. HISDL description of a D Master Slave Flip-Flop.

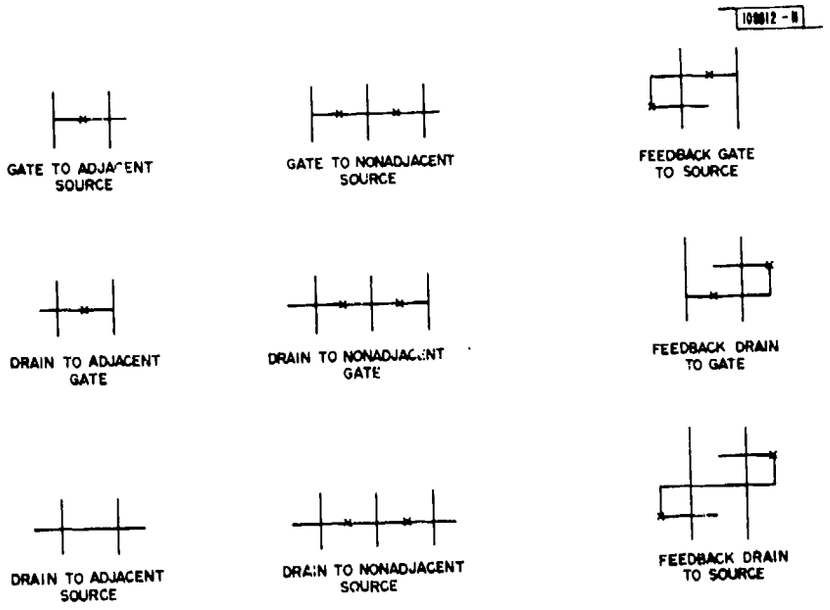


Fig. III-2. Nine interconnect cases.

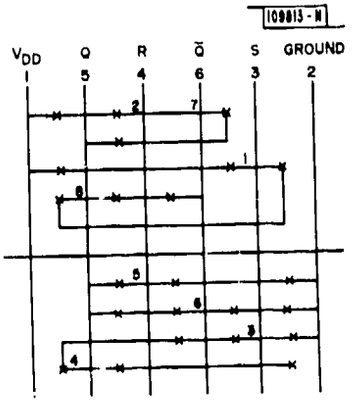


Fig. III-3. SR Flip-Flop layout without clusters.

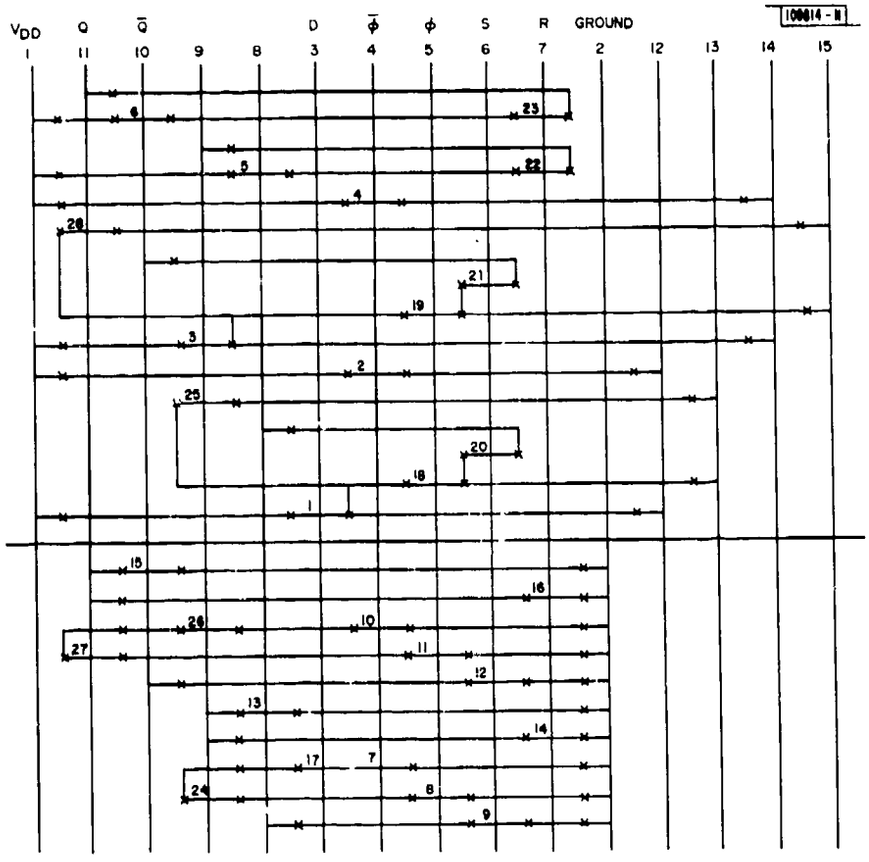


Fig. III-4. D Master Slave Flip-Flop layout without clusters.

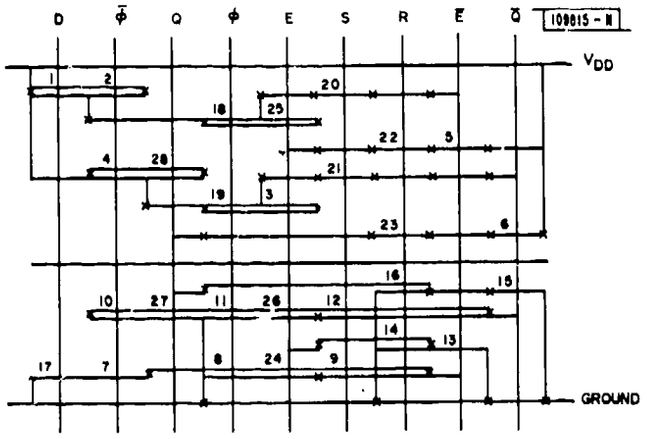
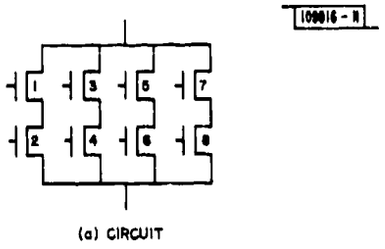
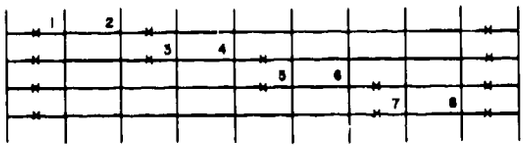


Fig. III-5. D Master Slave Flip-Flop.





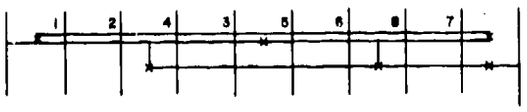
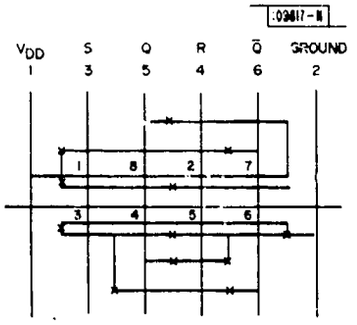


Fig. III-6(a-c). Pathological case showing benefits of clusters.

Fig. III-7. SR Flip-Flop layout using clusters.



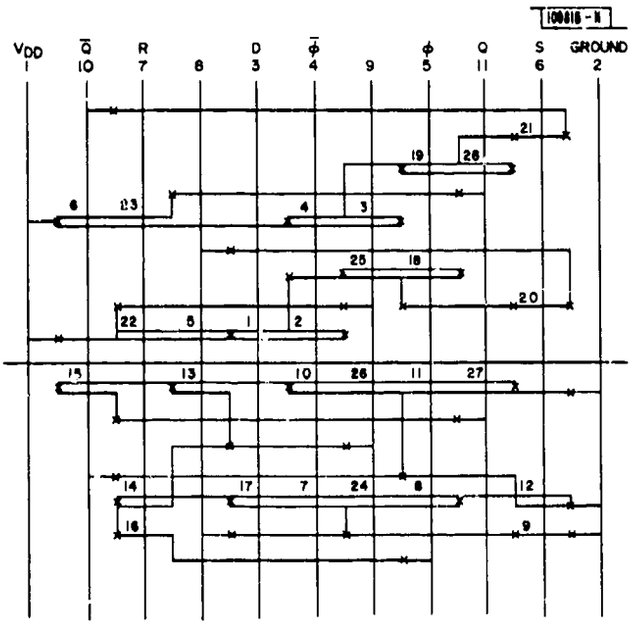


Fig. III-8. D Master Slave Flip-Flop layout using clusters.

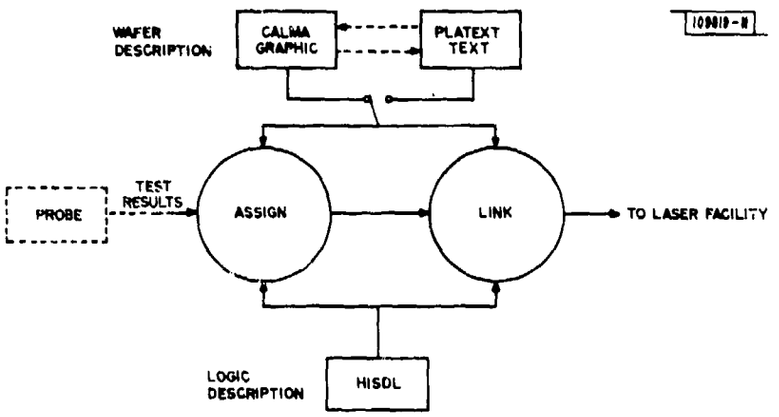


Fig. III-9. Conceptual diagram of PAL system.

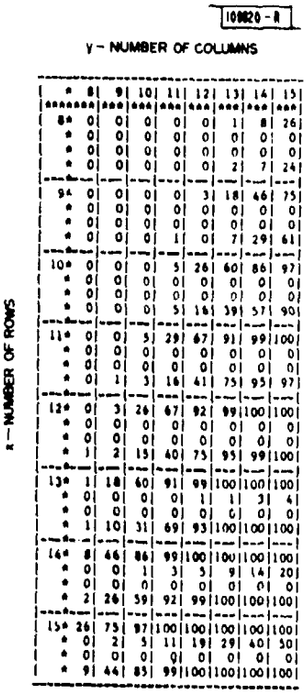


Fig. III-10. Wafer yields for an 8 × 8 integrator array mapped onto an x × y physical array for four restructurable interconnect constraints: (1) unconstrained, (2) unlimited SKIP, (3) SKIP(1), (4) BISKIP(1, 2); cell yield = 50 percent.

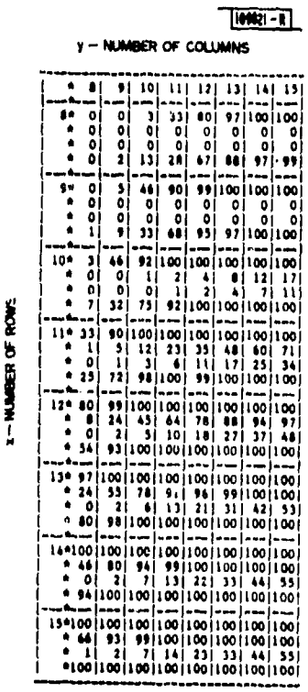


Fig. III-11. Wafer yields for an 8 × 8 integrator array mapped onto an x × y physical array for four restructurable interconnect constraints: (1) unconstrained, (2) unlimited SKIP, (3) SKIP(1), (4) BISKIP(1, 2); cell yield = 70 percent.

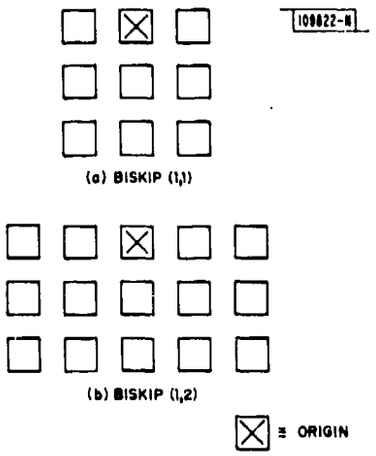


Fig. III-12(a-b). Cells reachable with two forms of BISKIP.

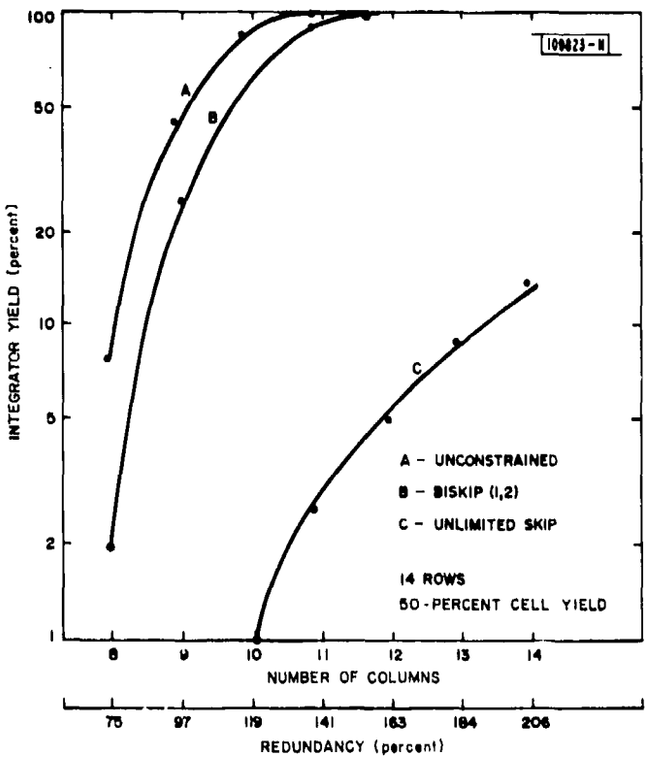


Fig. III-13. Integrator yield with three restructuring strategies.

IV. RVLSI TESTING AND APPLICATIONS

A. TESTING

1. Fault Detection

a. Issues in Test Vector Generation

There are three factors in formulating test vectors. From the bottom up, these are: how to test a gate, how to test a combinational network of gates, and how to address feedback (i.e., memory).

A fairly large body of literature exists where the failure set is restricted to "stuck at" types at the gate level. There is also some recent work on functional unit testing by means of a Boolean difference technique. A consequence of this latter method is that networks may not need to be considered.

In a network the gate inputs are generally not directly controllable, and the outputs not observable. However, techniques have been developed which determine how to set the controllable inputs so that the gate inputs conform to a given pattern and the output is propagated to an observable output. In addition, these methods allow reduction of the total number of vectors required by combining sensitized paths, common error cases, etc.

The only method for removing feedback from the circuit test appears to be the IBM level-sensitive scan technique.[15]

b. Problems with the Stuck-At Fault Technique

Some other readings seem to indicate that the stuck-at fault is not the most likely failure mode of NMOS, however. Most authorities claim that MOS failure modes have not changed from the days of MSI. The statistics for MSI are based on a study done by NASA in 1969. To summarize these statistics, about 50 percent of all defects are due to fabrication, and 50 percent to mounting and handling. Of all failures, 20 percent (or 40 percent of fabrication-related failures) are due to oxide defects. Another 20 percent are due to handling and overstress, some of which also manifest themselves as oxide (especially gate oxide) defects.

Figure IV-1 shows a CMOS NAND gate. It is assumed that the b input poly has punched through the oxide and has become shorted to the source diffusion of the N-channel gate. The output signatures of various possible faults are indicated in Table IV-1. Here, "D" is used to denote a certain fault. The "?s" in the "oxide flaw" column, which relate to the fault mentioned above, indicate that any combination of these might fail. The specific outcome depends on thresholds and drive capability of surrounding circuits.

Why is all this important? Obviously, any test for all the stuck-at type faults would reveal that something was wrong. However, in general, a full circuit contains many gates and signals. "Efficient" methods will attempt to find the smallest set of test vectors which cover all the expected (i. e., stuck-at) faults. This must be done by finding ways to combine cases and develop only one test condition where there are several which might reveal the same fault. The fact is that the oxide flaw case might not be discovered.

Further, there are several pathological possibilities. Suppose conditions are such that the oxide flaw case has the same signature as a-stuck-at-1. Because there are many other gates in the circuit, a test vector other than a = 0, b = 1 might be used to check for the a-stuck-at-1

TABLE IV-1
FAULT SIGNATURES FOR NAND GATE

	a	0	1	0	1
	b	0	0	1	1
	Correct	1	1	1	0
a	Stuck at 1	1	1	D	0
b	Stuck at 1	1	D	1	0
c	Stuck at 1	1	1	1	D
a	Stuck at 0	1	1	1	D
b	Stuck at 0	1	1	1	D
c	Stuck at 0	D	D	D	0
	Oxide flaw	1	?	?	?

case. The chosen test vector might easily set the b line to 0, so that nothing wrong would be observed at c. Even more likely, c probably is not directly observable. Since another path is being used to check for a-stuck-at-1, the test vector may completely mask the effect of c being wrong at subsequent gates. A final possibility is that the shorted gate effect is equivalent to two of the stuck-at faults. Since standard test vectors look for only single stuck-at failures, it is certainly possible that the chosen test vectors would totally conceal the fault.

On top of these general objections to the usefulness of standard test vector generation methods, the impact of RVLSI must be considered. From the VLSI standpoint, the standard "path-sensitization" method by which internal stuck-at faults are propagated to observable pins becomes more difficult, if not impossible, as chip complexity rises. In addition, redundancy hopelessly overcomplicates the situation for these testing methods.

c. Boolean Difference Technique

A test generation method other than the stuck-at scheme and its modifications is the Boolean difference technique.[16] This technique deals directly with a combinational functional block. The particular gate types comprising the block are unimportant. There are three steps to this technique:

(1) The partial Boolean differences of the function with respect to each input variable must be calculated.

(2) A pattern set is generated according to the critical paths.

(3) The test set is reduced by taking advantage of "don't care" inputs and merging compatible patterns.

The partial Boolean difference of a function f with respect to input x is another function, df/dx, which answers the question: Given constant values of the other input variables, will the output change value if the variable x is changed? For some patterns of the other inputs, df/dx may be 1; and for other patterns, 0. The set of vectors for which df/dx is one are the "sensitizing"

vectors for input x. Each critical path is tested by getting up each of the sensitizing vectors in turn and stimulating the x input. If the output responds, the path is good; if the output is constant, the path has failed.

Obviously standard sorts of analysis will yield "don't cares" to reduce the number of independent vectors in the test set. From Ref. 16, "... stuck-fault techniques produce a subset of patterns of the set the Boolean difference technique would produce. This is because the stuck-fault techniques would produce patterns which would test an input's ability to control the output for only one of the sensitizing conditions. The Boolean difference technique tests for all sensitizing conditions. It is for this reason that this technique retains its test quality when applied to any functional system." It should be added that it is less dependent on the fault model too.

Some might object that this quality is paid for by the requirement for much larger sets of test vectors. Two test cases are mentioned in Ref. 16: a 1-bit and a 4-bit ALU. At this level of complexity, the Boolean difference method generated a test set approximately 8X that of the stuck-at techniques for the 4-bit ALU, and the same test set for the 1-bit ALU. More points are needed for a realistic experimental complexity measure.

Notice that we have only checked that sensitized paths are, in fact, sensitive. Recall that oxide defects were our primary failure source. Although the oxide at poly-metal crossings is thicker, an observable defect would be a connection between the two lines, and to be rigorous we should also make sure that paths which are not supposed to sensitize an input, in fact do not. This would correspond to vectors where $df/dx = 0$. It is also possible that this would significantly increase the number of test vectors necessary.

d. Elimination of Feedback

In regard to the IBM scan technique, it seems possible to automatically incorporate the scan logic in a fairly straightforward way. By a method similar to the GAMMA program technique, the minimum set of feedback signals is identified. These signals are then isolated from the rest of the circuit by the scan logic. This method would surely be wasteful if there is a great deal of feedback in a functional design. It is not believed that this will be the case, however, because designs will probably be either memory intensive or function intensive (i.e., combinational) only. In cases where the designer introduces a small number of flip-flops into a generally combinational design, the automatic scan insertion overhead should be perfectly acceptable. Where the designer adds some gates into what is primarily memory, observing and modifying the state from the outside is usually very easy (i.e., by accessing the memory).

Memory, as such, should be tested by special rather than general methods anyway. The integrator is really a specialized memory-type cell with logic to perform certain types of restricted memory writes. As memory, the scan to/from the external world is already in place; test vectors for the combinational logic can be easily applied and observed.

e. Future Directions

There are many approaches to take based on the restriction of designing in MACPITTS (see Sec. III-A-3). MACPITTS compiles functional specifications into a combination of PLAs and register arrays. Since PLAs consist of two levels of NOR gates, there is considerable reduction in the complexity of determining the test vectors and in their number. More "active" testing techniques would augment MACPITTS to construct the PLA with extra signals and states so that the resulting functional unit was self-testing. One simple way this might be done is based on the same principle as Hamming codes for signal transmission.

2. Tester-On-Chip

a. Introduction

A crucial problem facing any integrated-circuit designer is testing. There are many approaches to testing and a variety of tester devices available. To better understand these approaches and develop a tester device for in-house use, the design of a minimal dynamic tester chip has been initiated (TOC for Tester-On-Chip). TOC was designed as an extension of a static tester as typified by the Stanford Minimal Tester (MPC 5/80). TOC leans in the same philosophical direction as the most-sophisticated commercially available testers.

b. Static vs Dynamic

The reason that the Stanford Minimal Tester can only test static devices is quite instructive. The simplest testers are limited to a maximum test rate by the speed of their interface. This is acceptable for static devices, where the state of the device under test is maintained indefinitely between inputs. Unlike static circuits, however, dynamic devices have a minimum speed of operation. The speed of a 9600-baud RS232 interface, which is standard on a VAX, is much too slow for testing even the most mundane dynamic circuit fabrication.

One possible modification would be to provide a memory which can be loaded with a sequence of test vectors. The vectors would then be applied at a reasonable speed. Another memory would record the results, which could then be transmitted back over the slower communication link. This, however, only postpones the problem; an arbitrary dynamic design would require infinite memory.

Another method might be to specify not the test vectors themselves, but some compact parameterization of them over the communication link. A compact specification could be a code or, in fact, a program. Many sophisticated testers use program-like elements, such as loops and complex conditional tests, to extend the virtual length of their highest-speed memories. However, even in these testers, vectors in the high-speed memory are eventually exhausted, and new ones must be transferred from a slower-speed backup. When this happens, the tester must output a small loop of test vectors which place and maintain the device under test in a hold loop. At the same time, the tester reloads its remaining memory space with new test vectors and then jumps to this sequence of outputs. A decision was made to construct a tester without general loop and conditional capabilities. The ability to output the hold sequence in a loop while reloading the rest of memory is included, however.

The decision to construct this minimal tester also implies a testability criterion that must be met by any device tested by TOC: all states of the device-under-test (DUT) must be situated on a trajectory between two hold loops. A hold loop is a series of states which repeat indefinitely under the repeated application of a hold sequence. The maximum trajectory length plus hold sequence is strictly limited by the size of the TOC memory.

c. TOC Design Decisions

Some further decisions have been made regarding the TOC design:

(1) Each TOC chip will drive or sense four (4) pins of a DUT and will contain the logic, but not the memory, needed for the tester. Memory will be supplied externally with commercial devices.

(2) TOC chips can be cascaded without limit. All TOC chips in the cascade connect to a single communication line. The cascade logic included on TOC regulates the use of the line.

(3) TOC will apply inputs to input pins and read outputs from output pins at a fixed frequency (unless stopped). There will be no hang-ups because of error detection or communication with the user.

(4) Each pin may be specified as input to the DUT, output from the DUT, or output from the DUT with check. The check value is specified by the user in the test vector. If a check fails, a status bit in TOC (readable by command over the interface) is set, and the user-supplied value is toggled in the vector memory. The vector memory can then be read back to the user over the interface for further processing. The total specification to each pin can change every TOC cycle.

Figure IV-2 shows TOC in its expected configuration.

d. Commands

The following commands can be transmitted over the serial communication line from external controller:

0-9, :-?	load "hex" value and increment test memory address
I	initialize, clear address registers and cascade bits
H	load next vector, start hold sequence at current address
N	enable next slice
G	go or continue
R	read status flag (YIN will be returned depending on errors detected)
S	stop
X	examine current memory location
null, <carriage return>, <linefeed>	ignore

e. Status

The control signals and data paths for the test memory address registers have been specified. A state diagram for the FSM controller has been drawn, and described in the MACPITTS language. Some simulation of the controller has been done. TOC's UART is composed of an FSM and register array also, and is in a similar state of development.

The current TOC is envisioned as being a "stand-alone" tester with the ultimate goal of using a later version as an on-wafer tester. TOC cells would be scattered around the wafer among "payload" cells and perform cell-level testing at any time, not just at wafer probe time. Many factors remain to be resolved, such as mechanisms for dynamic cell isolation during testing, and location of tester memory.

3. Interconnect Testing

One requirement of RVLSI is that both the cells themselves, as well as the interconnect, must be tested in order to configure a working system. According to the original conception, cells would be tested from the wafer periphery by way of the same interconnect as would be

used for cell-to-cell communication. Using reprogrammable links, each cell in turn would be connected alone to the external test pins. This approach has two shortcomings which make it unsuitable for our present efforts. First, it relies too heavily on the interconnect reliability and generality, by making cell testing dependent upon interconnect functionality. As we are not yet sure of expected interconnect yield, it is best to avoid using interconnect for testing. Second, it requires reprogrammable links, which must be included in the design in addition to the laser-zappable links. This adds complexity to the design.

The first step to the solution presented here is decoupling interconnect testing from cell testing. This is done by providing wafer probe pads at the pins of each cell for cell testing. This has the advantage that all cells may be fully tested on a wafer prober, separate from the laser zap table. As far as cell testing is concerned, a single pass of cell probe test, followed by zapping for configuration, is sufficient. If the laser configuration process does not have any adverse effect on either the cell yield or the interconnect yield, and is reliable, then this single-pass approach will be sufficient. All that remains necessary is a method for testing interconnect.

The interconnect test scheme proposed here is suitable for use only with laser zapping and arbitrary segmentation. Tracks must run the entire length of the wafer between two special test rails, to be described later. This implies that for a track to be testable it must be unbroken, thus requiring that segmentation be performed only after interconnect testing. The test scheme can be classified as interactive, in contrast to the single-pass scheme discussed above. An interactive test scheme incrementally zaps and tests the wafer in many short cycles. This requires that the laser zap table be equipped with testing facilities in contrast to a single-pass approach where all testing is completed before zapping commences. It seems that for interconnect testing, a single-pass approach is infeasible without providing an inordinate amount of interconnect probe pads and a suitable wafer-sized probe. The interactive interconnect testing scheme proposed here requires a minimum of test circuitry, just a continuity tester, at the zap table.

Several assumptions are made about the types of interconnect faults which are of two basic types: there can be a short between two tracks which should not be connected, and there can be a break disconnecting two paths which should be continuous. The scheme proposed here tests for three specific types of faults, namely: breaks in the continuity of wafer length tracks, shorts between two adjacent parallel tracks on the same interconnect layer, and shorts between two perpendicular tracks on different layers at their point of crossing. This last type of fault may occur when a link is accidentally shorted or may, in fact, be an intentional via in which case it would be a fault if no short were detected. Note that only adjacent tracks are tested for shorts under the assumption that nonadjacent tracks can be shorted only if the intervening tracks short as well.

Figure IV-3 shows a diagram of the configuration of the extra test rails used to support interconnect testing. In addition to the normal wafer length vertical and horizontal tracks, four test rails surround the perimeter of the wafer. These four test rails are terminated in eight probe pads labeled A through H. Special links are provided for connecting the interconnect tracks to the peripheral test rails. These are shown as small circles in Fig. IV-3. These special links allow a connection to be made and broken two times in succession. A link with this capability can be constructed from two, one-time make-break links in parallel. An example of such a link is the totem pole structure shown in Fig. IV-4.

The proposed interconnect can best be described by the algorithm of Fig. IV-5. The algorithm makes use of three special primitives – two to control the zap table, and one to perform continuity testing. The primitive zap-on(a,i,j) is used to turn on the j^{th} sub-link of the i^{th} special totem pole link along peripheral test rail a. The zap-off primitive functions in a similar fashion, turning off the link. The primitive con(x,y) checks for continuity between pads x and y, returning true if there is continuity and false if there is an open circuit.

It should be noted that failures reported by this algorithm do not always necessitate bypassing the faulty interconnect. If a shorted link is discovered where it is undesirable, an attempt can be made to turn off the link. It may even be possible that the configuration algorithms could make use of information indicating that certain links are already turned on and still find a suitable assignment and linking which make use of the faulty link. All the implications of this possibility have not yet been fully considered.

B. APPLICATIONS

1. Phase 0 Integrator

A version of the integrator has been laid out in the CMOS gate array technology. In addition to the 4 × 1 integrator slice, each gate array contains test logic to aid the design of clock circuits for this technology. The integrator was laid out on the Calma graphics system. The Calma encourages hierarchy by allowing cells to be defined and placed in larger cells.

Basic NAND, NOR, and pass-gate cells were designed and found to be useful. In addition, several other more specialized cells were designed. Each cell type is represented in the library as two mirror images because of the peculiar location of p-tubs in the gate array.

A major cause of inefficiency in using the gate array is gate pairing. Every p-channel/n-channel pair shares a common gate, so a pass gate must take two transistor pairs. This and the inefficiency of our rigidly enforced cell design rules are easily tolerable as the price of regularity and modularity. One potential design pitfall, however, is the p-tub layout of the gate array. A consequence of the p-tub layout is that connections forming a NAND gate in one row, form a NOR gate in the next, and vice versa. This is why mirror images for each cell are necessary, and some caution must be exercised during the process.

The wafer mask includes wafer-level interconnect. Restructurability is achieved by laser zapping. The amount of wafer-level interconnect is more than sufficient for connecting the integrator cells in all desired configurations. There is an upper limit to the amount of interconnect needed for defect avoidance. Interconnect requirements for testing, however, will not be known until the reliability of the laser-zapping processes is measured. To some extent, the reliability of the interconnect itself is unknown. These questions are to be resolved empirically.

Testing of the individual integrator cells is expected to be reasonably straightforward, since similar tasks have been undertaken in-house previously. Many more pins than are strictly necessary have been provided (such as inclusion of a "hold" signal) to facilitate testing. However, combined testing of cells and interconnect is a more complex problem. Some early schemes to test-while-linking have been proposed; however, more empirical data on the reliability of the laser interconnect are needed before moving in this direction. Considering that both the CMOS gate array process and the link technology are being debugged on the integrator design, a variety of unforeseen difficulties may be encountered in testing the entire structure.

2. Phase 1 Integrator

The circuit for the basic cell (four 10-bit counters input shifter and output selection logic) has been agreed on and laid out on the Calma system. Reaching this point involved computing capacitances in the various (wafer-scale) signal lines and sizing the CMOS transistors to fit. SPICE simulations were used to validate the assumptions.

Currently, a good deal of discussion is taking place regarding the format of the wafer-scale interconnect pattern. The issue seems to be meeting the need for distributing 25-MHz clock signals while maintaining a very regular pattern. The former requires very short leads so that capacitance is minimized, while the latter expedites the task of developing CAD tools which will perform the assignment and linking functions in a general way.

3. 3-4-5 Filter

In order to demonstrate the usefulness of the MOSIS facility, a simple nonrecursive digital filter was laid out and submitted for fabrication in January 1981. This filter is useful in pitch detection applications and is a relatively easy VLSI project since the "multiplications" are all by unity [Fig. IV-6(a-b)]. The filter was laid out from standard parts such as the pads and shift register cells from Xerox, PLAs, and wiring channels. This technique illustrates the style of sacrificing chip area and density in favor of easy (automatic) generation. The transfer function of the filter is:

$$H(z) = \prod_{k=3}^{5} \left\{ \frac{1}{k} \sum_{j=0}^{k-1} z^{-j} \right\} .$$

Another aspect of the 3-4-5 filter was the first attempt to use standard multiproject wafer pad configurations (see Sec. II-B-2). These pad frames are produced in four sizes – full die, half-die, quarter-die, and eighth-die. All but the smallest have 40 bonding pads in well-known positions. This feature permits post-fabrication testing using a small number of probe cards. Since all 40 pads are present whether or not they are used, automatic wirebonding is easy to implement. The ability to scribe the wafer into individual projects rather than multiproject chips allows better utilization of the wafer area. Finally, since each project can be individually packaged, no proprietary interests will be risked.

As it turned out, the 3-4-5 filter was slightly too big for the quarter-die pad frame. Placing it in the half-size frame allowed ample empty space which will be used for testing circuits if the filter is resubmitted. To date, only one of the five devices has been tested, and that one does not appear to work.

If it is submitted again, some provision must be made for specifying which of the 40 pins will be used for the substrate connection. The first five devices had 40 pads (30 active signals) and thus, at bonding time, it was not obvious which carrier pin was not needed and would be free for the substrate connection. The chip would need to have been studied in detail. In fact, the chip was mounted with a 90° rotation from what was expected, and the substrate connection is shorting one of the outputs. The Lincoln Laboratory wirebonding shop will redo the devices before further testing.

4. FFT Subsystems for Speech

A number of speech signal-processing algorithms use the FFT operation. Although a wider range of sizes is used, most employ a 256- or 512-point transform with a word size of at least 16 bits with, or without, dynamic overflow control. This study therefore assumes a 512-point transform with a 16-bit integer word and examines the capabilities and requirements of each implementation.

This study also makes several assumptions about the technology used. The functional logic will be CMOS with a maximum clock rate of about 20 MHz. The functional units will be arranged as cells with external (to the cell) restructurable interconnect. The cells may also incorporate internal restructurability. The basic cell size will be about 5 by 5 mm on a 3-in. (~75-mm) wafer. Assuming a redundancy of 2 and an area utilization of 0.5,

$$\frac{\pi(\frac{75}{2}\,\text{mm})^2}{(5\,\text{mm})^2} \times 0.5 \times 0.5 = 44 \text{ usable cells per wafer} \quad .$$

A basic element which will be assumed is a radix 2 butterfly cell. The cell will contain six bit-serial fractional multipliers, eight bit-serial adders, and eight 22-bit (tapped) shift registers. The adders and multipliers will have 2 and 32 clock cycle delays (first bit in to first bit out), respectively. Both will be capable of maintaining a full 1-bit-per-clock cycle data flow rate. Internal restructurability will be used to adjust the configuration of a cell as well as to avoid defects. Since a radix 2 butterfly only requires 4 multipliers and 6 adders, the extras are spares. The cell can also be configured to implement functions such as a second-order section or a multiply-accumulator.

A second cell might be a RAM or ROM where about 4 kbits (256 16-bit words) could be stored in the basic cell size. Special-purpose cells such as controllers will be suggested by each implementation of the FFT algorithm.

Several implementations of the (N = 512 point) FFT can be ruled out immediately. A full array of butterfly units would require $(N/2) \log_2 N = 2304$ butterfly cells. A single column of N/2 butterflies can implement a single stage (there would be $\log_2 N$ stages) of a constant geometry form of the FFT. This again can be ruled out as N/2 = 256 butterflies is far more than can be put on a single wafer.

A radix 2 pipeline FFT [Fig. IV-7(a-b)] is feasible. Each stage [Fig. IV-7(a)] consists of a commutator, a delay, a butterfly, coefficient generation, and a second delay. The commutator is a simple unit consisting of two multiplexers and a state flip-flop. The delays are shift registers. Coefficient generation by a simple recursion in a modified butterfly unit is compact and simple. Due to the multiplier delay, however, only one coefficient per 2 word times (1 word time = 16 clock cycles) can be generated, thus halving the pipeline throughput. A ROM supporting the full throughput could also be used, but either modularity or a significant amount of area would be sacrificed. (The required ROM size for each stage would be M complex words or 8192 bits requiring two cells for the largest. Either the ROM size must scale with each stage, or several standard size ROMs with unused locations could be used.) The delays might be modularized by designing one large restructurable shift register that could be sliced into the desired size pieces with jumpers around any defects.

Overall, the pipeline FFT would require $2 \log_2 N = 18$ butterfly units (or 9 butterfly units and at least 16 kbits of ROM), 9 commutator units, about (3/2) N words = 12 kbits of shift

register, and a reasonably simple controller. I/O would be serial with the output in bit-reversed order. Dynamic overflow control is not easy to implement within this structure. The overall time to compute an FFT would be about 1.4 ms (first word in to last word out), or 0.7 ms with ROM coefficient generation. However, two FFTs could be performed simultaneously.

Several "single butterfly unit" FFT architectures [Fig. IV-8(a-b)] are feasible. All require an N complex word RAM (= 16 kbits = 4 cells) if in-place, or two "ping pong" RAMs (= 32 kbits = 8 cells) if not in-place. Coefficient generation would be by an N/2 complex word ROM (= 8 kbits = 2 cells). The controller would consist of three nested counters and a set of adders for address generation, exactly as one would organize a typical software implementation. Overflow control can easily be implemented with one guard bit, a "potential overflow on the next column" detection circuit (i.e., an exclusive OR of the two most-significant bits of the butterfly outputs), and a right shift-on-read function on the RAM. Bit reversal could occur on the input or output operations. This class of FFT implementations would require about 3 ms, including I/O to perform an FFT. Since it is not modular, defect avoidance by restructurability would be possible only by duplication of each unit.

In summary, the pipeline implementation and a set of "single butterfly" implementations appear to be feasible. The pipeline implementation is faster and more modular. However, it is larger, cannot control overflow easily, and has a bit-reverse ordered output. The "single butterfly" implementations are smaller, can accommodate overflow control, and have normally ordered I/O. However, they are slower, have more complex controllers, and are not modular. Other radix implementations, while not specifically examined here, appear to offer little advantage over the radix 2 systems for speech applications.

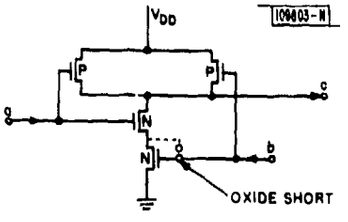


Fig. IV-1. CMOS NAND gate.

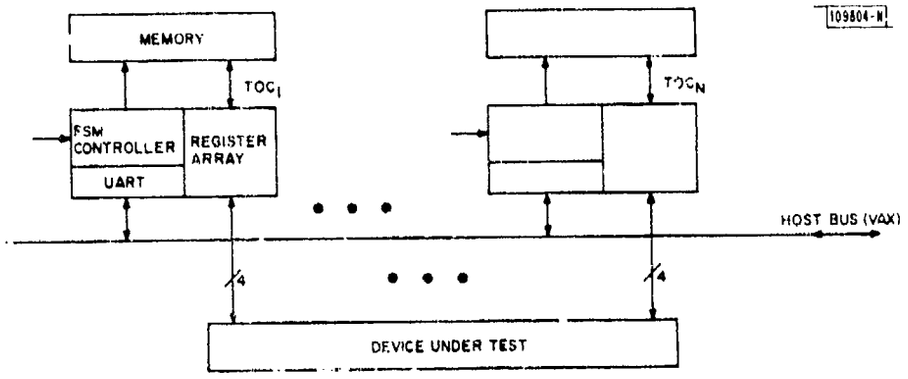


Fig. IV-2. Tester-On-Chip (TOC).

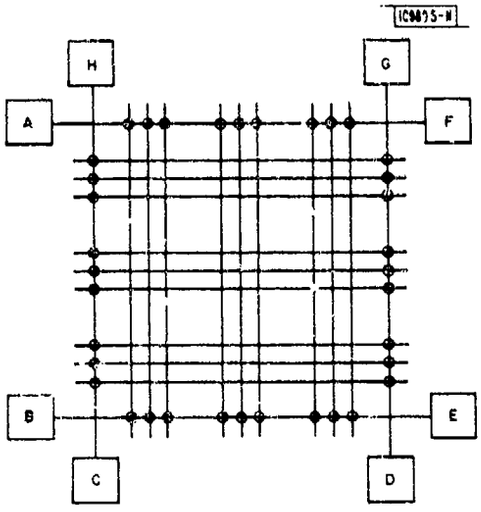


Fig. IV-3. Testable interconnect pattern.

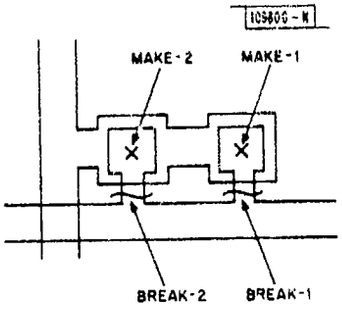
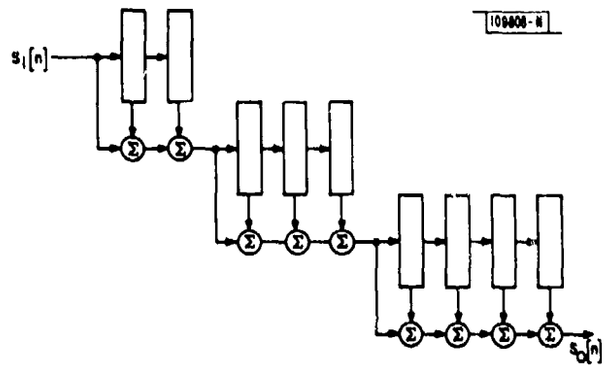


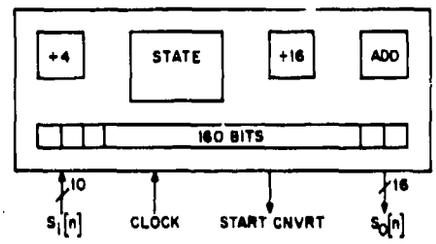
Fig. IV-4. Totem pole twice programmable link.

```
procedure test-interconnect                                    108807-5

/* initial test of test rail continuity */

  if not con(A,E) or not con(B,F) or
     not con(C,I) or not con(D,G)
  then /* wafer is not testable */ fi;

/* test for breaks in horizontal tracks and for shorts
   between adjacent horizontal tracks */

  for I from 1 to number-of-horizontal-tracks
  do if I = 1
     then map-on(C,1,1)
     else map-off(D,I-1,1) fi;
     map-on(D,I,1);
     if not con(C,D)
     then /* horizontal track I is broken */ fi;
     map-off(C,I,1);
     if I = number-of-horizontal tracks
     then map-off(D,I,1)
     else map-on(C,I+1,1);
          if con(C,D)
          then /* short between horizontal tracks I and I+1 */ fi fi od;

/* test for breaks in vertical tracks and for
   shorts between adjacent vertical tracks */

  for J from 1 to number-of-vertical-tracks
  do if J = 1
     then map-on(A,1,1)
     else map-off(B,J-1,1) fi;
     map-on(B,J,1);
     if not con(A,B)
     then /* vertical track J is broken */ fi;
     map-off(A,J,1);
     if J = number-of-vertical-tracks
     then map-off(B,J,1)
     else map-on(A,J+1,1);
          if con(A,B)
          then /* short between vertical tracks J and J+1 */ fi fi od;

/* test for shorts between crossings of horizontal and vertical tracks */

  for J from 1 to number-of-vertical-tracks
  do map-on(A,J,2) od;
  for I from 1 to number-of-horizontal-tracks
  do map-on(C,I,2);
     if con(A,C)
     then /* horizontal track I shorts to some crossing vertical track */ fi;
     map-off(C,I,2) od;
  for J from 1 to number-of-vertical-tracks
  do map-off(A,J,2) od;

  for I from 1 to number-of-horizontal-tracks

  do map-on(B,I,2) od;
  for J from 1 to number-of-vertical tracks
  do map-on(D,J,2);
     if con(B,D)
     then /* vertical track J shorts to some crossing horizontal track */ fi;
     map-off(D,J,2) od;
  for I from 1 to number-of-horizontal-tracks
  do map-off(B,I,2) od;

  end test-interconnect;
```

Fig. IV-5. Interconnect test algorithm.

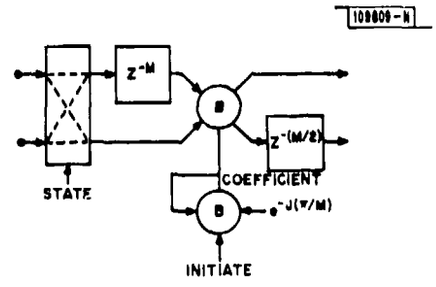


(a) BLOCK DIAGRAM



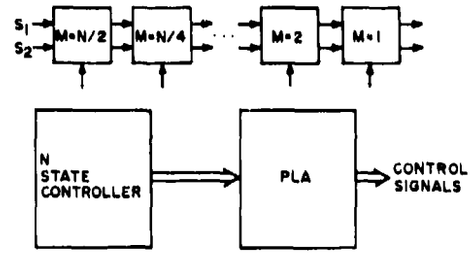
(b) CHIP STRUCTURE

Fig. IV-6(a-b). FIR low pass filter for pitch tracking.



(a) PIPELINE STAGE

Fig. IV-7(a-b). Pipeline FFT.



(b) THE PIPELINE

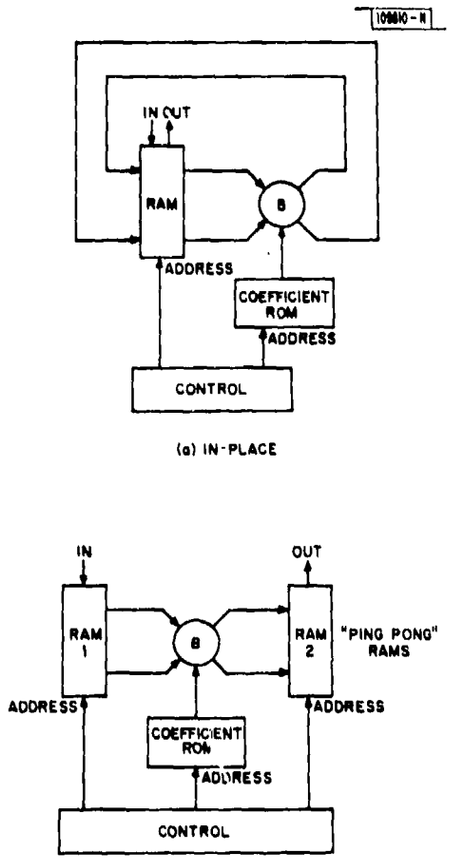


Fig. IV-8(a-b). Single butterfly FFTs.

REFERENCES

1. L. Conway et al., "Implementation Documentation for the MPC79 Multi-University Multiproject Chip-Set," Xerox PARC document (1 January 1980).
2. This was proposed by Alan Bell, Xerox PARC, at the May 1980 ARPA VLSI contractors meeting.
3. J. I. Raffel, "On the Use of Nonvolatile Programmable Links for Restructurable VLSI," Proc. California Institute of Technology Conf. on Very Large Scale Integration, January 1979, pp. 95-101.
4. Semiannual Technical Summary, Restructurable VLSI Program, Lincoln Laboratory, M.I.T. (31 March 1980), DTIC AD-A096075.
5. C. A. Mead and L. A. Conway, Introduction to VLSI Systems (Addison-Wesley, Reading, Massachusetts, 1980).
6. L. Conway et al., "MPC79: The Large-Scale Demonstration of a New Way to Create Systems in Silicon," Lambda 1, 10-19 (1980).
7. T. Strollo et al., "Documentation for Participants in the MPC580 Multiproject Chip-Set," Xerox PARC document (7 July 1980).
8. J. Raffel et al., "Laser Programmed Vias for Restructurable VLSI," Proc. IEEE Intl. Electron Devices Mtg., Washington, DC, 8-10 December 1980, pp. 132-135.
9. W. Plummer, "Chip Maker's Guide," Lincoln Laboratory internal memorandum (12 December 1980).
10. A. D. Lopez and H-F. S. Law, "A Dense Gate Matrix Layout Method for MOS VLSI," IEEE J. Solid-State Circuits SC-15, 736-740 (1980).
11. K. H. Konkle, "Gate Matrix Style for Our CMOS Process," Lincoln Laboratory Group 23 VLSI internal memorandum (1 June 1980).
12. C. Hoare, "Towards a Theory of Parallel Programming," in Operating Systems and Techniques (Academic Press, New York, 1972), pp. 61-71.
13. J. Siskind, "PLATEXT Component of the PAL CAD System," Lincoln Laboratory internal memorandum (2 June 1980).
14. A. Giovinazzo, "A Mask Design Rule Checking System," Proc. IEEE Intl. Conf. on Circuits and Computers, Port Chester, New York, 1-3 October 1980, pp. 932-936.
15. E. Eichelberger and T. Williams, "A Logic Design Structure for LSI Testability," Proc. 14th Conf. on Design Automation, New Orleans, Louisiana, 20-22 June 1977, pp. 463-468.
16. R. A. Todd, "A Test Generation Technique for Devices without Gate-Level Descriptions," M.I.T. VLSI Memorandum 80-23, Department of Electrical Engineering and Computer Science, M.I.T. (July 1980).

GLOSSARY

CAD	Computer-Aided Design
CIF	Caltech Intermediate Form
CMOS	Complementary Metal-Oxide Semiconductor
CSP	Communicating Sequential Process
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
FSM	Finite State Machine
HISDL	Hierarchical and Iterative Structure Description Language
LICL	Lincoln Integrated Circuit Language
MDRC	Mask Design Rule Checker
MNOS	Metal-Nitride-Oxide Semiconductor
MOS	Metal-Oxide Semiconductor
MPC	Multiproject Chip
MSI	Medium-Scale Integration
PAL	Placement, Assignment, and Linking
PLA	Programmable Logic Array
RVLSI	Restructurable Very Large Scale Integration
TOC	Tester-On-Chip
VLSI	Very Large Scale Integration

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-81-153	2. GOVT ACCESSION NO. AD-A108 276	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Restructurable VLSI Program		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Summary 1 April 1980 – 31 March 1981
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Peter E. Blankenship		8. CONTRACT OR GRANT NUMBER*(s)* F19628-80-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order 3797 Program Element No. 61101E Project No. 1D30
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE 31 March 1981
		13. NUMBER OF PAGES 64
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB Bedford, MA 01731		15. SECURITY CLASS. *(of this report)* Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

VLSI	customization	routing
Restructurable VLSI (RVLSI)	hardware description language	systolic array
programmable interconnect	placement	integrator
defect avoidance		

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes work on the Restructurable VLSI Research Program sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency during the two semiannual periods, covering 1 April 1980 through 31 March 1981.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*